Data Analysis and Machine Learning:

on Long Memory Commodity Time Series

Tianyuan Ni
Supervisor: Dr. Hirbod Assa
Department of Mathematical Sciences

Thesis submitted for the degree of
Master of Philosophy
University of Liverpool

2019

Acknowledgements

Firstly, I would like to express my deepest gratitude to my supervisor Dr Hirbod Assa, at Department of Mathematical Science of the University of Liverpool, for providing me with the opportunity to embark upon an MPhil, and his precious guidance, feedback, continuous support and motivation in the project. I also want to show appreciation to my second supervisor Dr Leonardo Rojas Nandayapa at Department of Mathematical Science of the University of Liverpool and Dr Mehdi Madoliat, at Department of Mathematics of Marquette University for acting as mentors and providing valuable advice in this thesis.

I am also thankful to the whole Department of Mathematical Science at the University of Liverpool, including all the academic staff, postgraduate students and friends I have met here in both Liverpool and London campus, for providing help for answering all my questions and enjoyable chats.

I would like to take this opportunity to thank Dr Matthias Helmbold, Head Technical & Services at MAXIS Global Benefit Network for sharing his experience and knowledge in the real insurance and reinsurance industry. I am extremely grateful, in particular, his continued support, guidance and interesting discussion about innovative applications of my research.

Lastly, My special thanks extend to my family, especially my parents who supported me emotionally and financially. I am also grateful to my husband for his love and endless encouragement throughout all my research experience.

Abstract

Thesis Title: Data Analysis and Machine Learning: on Long Memory Commodity Time Series
Author: Tianyuan Ni

Recently, data driven approaches to modeling time series become important in today's market environment. This thesis studies the application of machine learning techniques on long memory financial time series, motivated by two main challenges that are attributed to the long memory and economic variables dependence structures.

Our research is to develop a data analysis framework including 1) Examine data characteristics and stylized facts, in particular, long memory modeling with skewed and kurtotic shocks of commodity daily price data, 2) Perform cluster to find proper groups that commodities are related in both the time and frequency domains, as well as to reduce their dimensionality, 3) Develop preferred neural networks to predict future values for commodities prices.

Our study demonstrates that the majority of commodity time series exhibit long-memoryness, hence the linear ARIMA model with stable shock is able to model the data. Model-based functional data clustering on both time and frequency domain can identify different products into groups on a reduced dimension. Lastly, by taking into account the cluster results on long memory features, our novel recurrent dynamic neural network forecasting algorithm can provide stable and high accurate forecast results.

Conclusively, our machine learning techniques combining financial time series clustering and forecasting algorithms outperform traditional theories and existing models, which can improve the model performance in commodity time series analysis and inform better investment decisions as well as risk management.

Contents

Chapter 1

Introduction

1.1 Motivations

The mounting volume of data and rapid development of technology keep transforming the financial market, thus modeling and forecasting of financial time series are essential when adopt data analytics techniques to inform better investment and risk management decisions. Because of the huge and complex sets of random technical indicators that drive the dynamic of prices, although today's technology has great potential to bring together big data and other relative information and manage informative data pictures more accurately, we still lack an efficient forecasting model to capture the majority data features and variation tendency.

The Efficient Market Hypothesis (EMH) suggests that the prices on market fully reflect all the available information, while, a growing number of researches reject the EMH showing that predictive properties do exist in financial markets [118, 123, 52, 76]. It is well known that price data often contain a lot of noise and complex relationships which do not follow a pure random walk, and price changes are neither independent nor identically distributed.

Therefore, whether predictions can be made about future values of financial time series is of significant interest. In general, there are two main challenges existing in financial market:

1) Long memory in financial time series. There are various linear and nonlinear dependencies between successive price changes, so that the classical time series models like ARMA process are no longer valid to forecast future

behavior as they cannot capture both the nonlinearity and the long-range dependence features into a single time series model. The existence of long memory in financial markets has been an important subject of both theoretical and empirical studies since around 1980. If assets display long memory, the series realizations may appear to have nonlinear properties and are not independent over time. This observations of persistence in time series mean that a slower decay of the autocorrelation function would be expected, and values from the remote past can help to forecast future behavior.

2) Dependence on economic variables. Another feature of financial time series is that they are usually multi-dimensional, so that one can have higher dimensional vector time series representing yet-to-be-executed prices for each instance of the time series. It has been observed that a wide range of economic variables exhibit dependent structure, so the study of correlation coefficients between variables plays a key role in the high dimensional time series analysis. Given several time series, the dependence is the measuring of the existing dependency among different variables, or, the dependency of exogenous factors. There are several measures available in economics such as correlation and copulas. While these approaches have their own advantages and disadvantages, there is no clear view on how both methods differ over time and how we can identify similarities and dissimilarities among these variables.

Although many economic data can be adequately explained by linear mathematical models, they can usually be better modeled using nonlinear models. An attractive way is to make minimal assumptions and use a data-driven, model-free and nonparametric approach. This motivates the use of machine learning for data clustering and financial forecasting.

There are four main markets in the financial world: bond market, stock market, commodity market and exchange market. This thesis discusses and addresses some of the difficulties associated with practical data analysis with real-world commodity market, followed by time series analysis within three parts: characterization, clustering and forecasting. A new data analysis framework is developed based on various machine learning algorithms to efficiently detect structures and optimal features in commodity data, in order to have better forecast results.

1.2 Chapters Overview

In this thesis, we start with introduction, which includes the motivations and objectives for undertaking the research. From the literature review of efficient market hypothesis to the long memory time series process, we study some important stylized facts of commodity daily index and future prices, with a detailed description of the data set shown in the second section. Our study shows that most of the price processes have unit roots with stationary, skewed and kurtotic increments and long-range dependence, therefore the classical normality assumptions with constant volatility are not appropriate for modeling the commodity daily index and futures prices data in today's real-world financial market.

In the second chapter, we continue further study on the long memory and ARFIMA model. The model of an Autoregressive Fractionally Integrated Moving Average (ARFIMA) process is used to model a time series with long-range dependence, or long memory. We first discover that suitable models for commodity daily index and future prices are ARFIMA(p, d, q) models with skewed and kurtotic (e.g., α-stable) shocks. Since non-linear autoregressive models are practically very difficult to implement, we study another measure of fractionally of processes - generalized Hurst exponent. We show that if we focus our attention to this measure, rather than the R/S Hurst exponent, linear ARIMA$(p, 1, q)$ models with $\alpha-$stable shock can still be good candidates for modeling commodity data.

In order to capture the dynamics of the data, the third chapter consists of multidimensional data clustering for commodity futures on both time domain and frequency domain. Following the basic clustering methods, the model-based functional data clustering methods are developed on time domain, to approximate density of functional random variables. On the frequency domain of the long memory process, the clustering modeling simultaneously estimates spectral density functions for stationary time series with some common features. We then present the experimental results of the two clustering methods.

The fourth chapter implements a powerful recurrent dynamic neural network forecasting algorithm that enables us to predict future prices of a series of commodity data, while taking into account the clustering results on long memory features of the financial instruments. In order to evaluate the clustering methods in terms of prediction performance, a comparison to the relevant machine learning models is given. Even though the framework of different clustering

algorithms is designed for long memory time series, one can build a forecasting algorithm providing the smartest features to have consistent and unbiased estimates of futures values.

The last chapter concludes the thesis and highlights the contributions. At the end, potential developments for further work are addressed.

1.3 Efficient Market Hypothesis

The basic and primary hypothesis about the market behavior is the efficient market hypothesis (EMH), a well-known theory in financial economics. From the 1960s, it was generally believed that stock markets were extremely efficient in reflecting information, prices must always show a full reflection of all available and relevant information and should follow a random walk process [38]. In other words, under the efficient market hypothesis (EMH), a market is said to be efficient with respect to an information set if the price fully reflects that information set.

Based on the information set, there are three common types of efficient markets:

- Weak form efficiency: The information set includes only historical prices.
- Semi-strong form efficiency: The information set includes all publicly available information.
- Strong form efficiency: The information set includes all public and private information.

Market efficiency concludes that martingales is able to model prices, and there should be no profit opportunity by using historical data since the best forecast of the future price is the current price information. Cowles [30] suggested that professionals cannot earn investment returns in the efficient market. Working [122] also discovered that it is impossible to predict price changes successfully in an ideal futures market. Thus, if all relevant information is immediately incorporated into current prices, future prices cannot be predicted by analyzing prices from the past. In other words, changes in prices should be independently and identically distributed. However, under the efficient market hypothesis (EMH), investors' trading strategies are correlated due to the arbitrage [40]. Mandelbrot [88] proposed that successive changes in prices are not independent Gaussian

random variables, but rather exhibit many recognizable patterns and are non-stationary. Meanwhile, Cootner [29] found that the stock market does not follow a random walk. Forecasting price or return ratio became more challenging and it was difficult to reconcile with the standard hypothesis. From the 1980s, many studies started to concern the EMH. Grossman and Stiglitz [49] showed that it is impossible for a market to be perfectly informationally efficient and prices cannot perfectly reflect the information that is available. A number of studies focused on the test and validation of the weak form efficient market hypothesis (EMH) with respect to stock markets, concluding that most of the financial markets are either weak form efficient or inefficient (see [118, 123, 52, 76]).

In the real financial market, economic and financial historical data typically exhibit some distinct low-frequency, non-periodic cyclical patterns. This is called a long memory phenomenon which was first discovered by Granger [45]. Most of the financial time series, including stock returns and commodity prices appear to be long-range dependent (LRD) (see [48, 55, 80, 96]). Mandelbrot [86] summarized that in the presence of long memory, pricing derivative securities with martingale methods might not be appropriate. If asset returns display long memory, the series realizations are not independent over time, and values from the remote past can help forecast future returns. Therefore, the presence of long memory in asset returns indicates that by conditioning on historical returns, future asset returns can be predicted more accurately.

1.4 Time Series Modeling

A time series is a sequence of data points over a time interval. Many sets of data are time series, such as daily stock price, monthly house price, yearly global temperature and so on.Time series analysis on scientific applications, particularly in the field of economics and finance, as well as the physical and biological sciences, has revolutionized the way people handle a sequence of data set.

Assume a time series can be defined as a collection of random variables over time. Usually, we have collected data (observed sample) beginning at some particular date and ending at another (say, from $t = 1$ to $t = T$)

$$\{y_1, y_2, \ldots, y_T\}.$$

A time series $\{y_t\}_{t=-\infty}^{\infty}$ is defined by describing the time series variable y

observed at time t, where t is typically discrete and varies over the integers. y_{t-1} is the value of y in the previous period and called its first lagged value. Similarly, the jth lag is denoted by y_{t-j} or L^jy, with L the log operator such that $L^i y = y_{t-i}$ for $i \in \mathbb{R}$, and the value of y that is h period ahead is denoted by y_{t+h} or $L^{-h}y$. Time series analysis involves methods for analyzing data in order to extract statistics of the data. In this section, we will discuss four classic time series models that are useful for the prediction of future values based on observed values.

From the time domain approach, the landmark work from Box et al. [17] developed a systematic class of models called Autoregressive Moving Average (ARMA) models. Such models are to explain the structure of the current value on past values. An ARMA(p, q) process is defined as a combination of the autoregressive process (of order p) and moving average process (of order q)

$$y_t = \mu + a_1 y_{t-1} + a_2 y_{t-2} + \cdots + a_p y_{t-p} + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \cdots + \theta_q \varepsilon_{t-q}$$

where ε_t is white noise process of random variables that are usually assumed to be independent and identically distributed and $a_i \in \mathbb{R}$, $\theta_i \in \mathbb{R}$ are the parameters for autoregressive and the moving average terms. Also, $\mu \in \mathbb{R}$ is normal a constant. Note, an ARMA(p, q) process can be defined as

$$\phi(L) y_t = \theta_0 + \theta(L)\epsilon_t,$$

where $\phi(L) = 1 - \sum_{i=1}^{p} a_i L$, $\theta(L) = 1 - \sum_{i=1}^{q} \theta_i L$ are polynomial operators in L of degrees p and q and $\theta_0 = (1 - \sum_{i=1}^{p} \phi_i)\mu$. For simplicity, one can also denote $\bar{y_t} = y_t - \mu$ and have the ARMA model presented as

$$\phi(L)\bar{y_t} = \theta(L)\epsilon_t.$$

According to Box et al. [17], an ARMA process is only stationary if the roots of $\phi(L) = 0$ lie outside the unit circle. The process exhibits explosive nonstationary behavior if the roots lie inside the unit circle. However, when the roots are on the unit circle, to achieve stationary, one can rewrite the nonstationary ARMA(p, q) process as

$$\phi(L)(1 - L)^d \bar{y_t} = \theta(L)\epsilon_t$$

so that the autoregressive operator $\phi(L)$ can be stationary again for some value

d which we will discuss further in later chapters, and we call such processes autoregressive integrated moving average ARIMA(p, d, q) processes.

Remark 1. A process $\{y_t\}_{t=0,1,2,...}$ is an ARIMA$(p, 1, q)$ process if $\phi(L)$ has only one unit root $\lambda = 1$ and the increment process $\{y_t - y_{t-1}\}_{t=0,1,2,3...}$ is stationary.

In addition, time series has been widely used for model and prediction purpose. For example, McCleary et al. [94] applied the ARIMA model to be the baseline building block for impact assessment, forecasting, and causal modeling. Kalpakis et al. [66] demonstrated the desired features of LPC cepstral coefficients based on accurate clustering and efficient modeling of ARIMA time series, while Hamilton [51] proposed this type of model for demonstrating changes in regime based on U.S. real GNP data, which can also be used as an objective criterion for measuring U.S. economic cycle.

1.5 Commodity Data Description

A commodity is a basic good that can be bought and sold or is interchangeable with other products of similar value. Some examples of commodities include grains, gold, beef, gold and oil. However, trading commodities is much more complex than simply buying and selling goods on the spot market. Derivatives such as Futures, Forward, Swaps, Exchange-traded Commodities (ETC) have become the primary instruments in commodity markets. In our study, we focus on the commodity futures and index. A commodity futures contract is an agreement to buy or sell a predetermined amount of a commodity at a specific price on a specific date in the future. A commodity index is a fixed-weight index or (weighted) average of selected commodity prices, to measure commodity price and investment return performance. The index values usually vary according to underlying commodities and are often traded on exchanges.

The commodities in our study are classified into five main categories: grains, softs, metals, meats and energy. The daily commodity index and futures data are obtained from Bloomberg [1] and we focus on the most actively traded commodities in each category. Thus, a total of eighteen different commodities are studied, namely Grains: wheat, soybean oil, soybean, oats, corn, canola; Softs: sugar, orange juice, cocoa, coffee; Metals: aluminum, gold, copper, palladium,

[1] Bloomberg Professional. Retrieved from Bloomberg terminal.

platinum; Livestock: lean hogs, feeder cattle; Energy: crude oil. For the univariate analysis, we use the closing price, which is the last observed price traded on a given trading day.

In the Appendix, Table 6.1 provides a brief description of the data sources of commodities. The first column lists the varieties of commodities and the second column briefly presents data sources. NYMEX refers to New York Mercantile Exchange, LME: London Metal Exchange, CME: Chicago Mercantile Exchange, NYBOT: New York Board of Trade, ICE: Intercontinental Exchange, WCE: Winnipeg Commodity Exchange. The third column lists the period of commodity prices data, as we can see most commodities are available from the 1970s or 1980s. In our study, we choose the common time period from 01 January 2008 to 31 August 2018, and a total of $100,224$ daily observations are examined.

1.6 Stylized Facts

Generally, stylized facts are widely understood to be an economic term defined by empirical findings that are accepted as truth, for example, the common properties that can characterize random varieties of financial instruments and markets (see [25, 53, 15]). Stylized facts tend to be values that are presentations of the data sets' empirical characteristics. Such measurements of data properties are usually model independent and are calculated from the data itself directly (see [34, 28]). In order to build the framework to analyze, cluster and forecast, it is important to understand the underlying fundamentals and empirical properties of commodity times series.

In this section, we study four basic stylized facts of the daily index and future commodity prices: stationarity, dependence, skewness and kurtosis. Most of the financial price time series, including commodity prices, appear to have unit roots with stationary increments, long-range dependency, skewness and kurtosis with different fractional characteristics.

Stationarity

Definition 1. A time series $\{y_t\}$ is stationary if for any sequence $s_1, \cdots, s_n \in \mathbb{N} \cup \{0\}$ and natural number $t \in \mathbb{N}$, $(y_{s_1}, \cdots, y_{s_n})$ has the same joint distribution as $(y_{s_1+t}, \cdots, y_{s_n+t})$.

Definition 2. A time series $\{y_t\}_{t=0,1,2,...}$ is stationary in weak sense if $y_t, \forall t \in \mathbb{Z}$ has a finite variance and

1. $\exists \mu \in \mathbb{R}$ such that $E(y_t) = \mu$ for all $t \in \mathbb{Z}$,
2. $E[(y_{t_1} - \mu)(y_{t_2} - \mu)] = \gamma(|t_2 - t_1|)$ for $t_1, t_2 \in \mathbb{Z}$

We show that all commodity time series except few are unit root processes with stationary increments. In Appendix Tables 6.2-6.4 we give the values of unit root test for each commodity. We use Augmented Dickey-Fuller (ADF) test and Kwiatkowski-Phillips-Schmidt-Shin (KPSS) test for the unit root tests.
The intuition behind ADF test is that it determines how strongly a time series is defined by a trend. The null hypothesis of the test is that the time series can be represented by a unit root, in other words, it is not stationary. The alternate hypothesis (rejecting the null hypothesis) is that the time series is stationary. We interpret this result using the p-value from the test. A p-value below a threshold (such as 5% or 1%) suggests we reject the null hypothesis (stationary), while a p-value above the threshold suggests we fail to reject the null hypothesis (non-stationary). Please note, a minimum p-value of 0.001 is set in the ADF test package we used from the Python package *statsmodels*.
On the other hand, the intuition behind KPSS test is that it determines how strongly a time series is defined by a trend. The null and alternate hypothesis for the KPSS test are opposite that of the ADF test. Same as ADF test, we interpret this result using the p-value from the test. A p-value below a threshold (such as 5% or 1%) suggests we reject the null hypothesis (non-stationary), otherwise a p-value above the threshold suggests we fail to reject the null hypothesis (stationary). Please note, a maximum p-value of 0.1 is set in the KPSS test package we used from the Python package *statsmodels*. In these tables, "ADF_P" is the test statistic $p-$value for ADF test. "$KPSS_P$" is the p-value from the $KPSS$ test. "Lags" are the lags for the time series model which is defined in Definition 4. As one can see, except oats index and gold index prices, all other time series have unit roots.

Dependence

In statistics and time series analysis, the autocorrelation function (ACF) of a time series process y_t describes the correlation between values of the process at different times.

Definition 3. The autocorrelation function (ACF) as a function of the two times or of the time lag h can be calculated as

$$ACF_h = \frac{\sum_{t=h+1}^{T}(y_t - \bar{y})(y_{t-h} - \bar{y})}{\sum_{t=1}^{T}(y_t - \bar{y})^2}$$

In addition to the ACF, the partial autocorrelation function (PACF) studied in our research gives the partial correlation of a time series with its own lagged values.

In Appendix, Figure 6.3-6.8 give the sample ACF and PACF patterns for the daily price data for all products. We can see for all products, the autocorrelation decays to zero very slowly, which is consistent with the observation from unit root test. The first lag value of partial autocorrelation function is statistically significant and close to one, whereas partial autocorrelations for all other lags are not statistically significant.

Recently, many research have shown that macroeconomic data presents properties from long-range dependence time series processes, and similar to our sample ACF results, the rate of decay of statistical dependence is slower than an exponential decay. Long-range dependence (or long memory, long-range persistence) has proven to be very important in modeling financial and economic data sets (see [79, 45, 104]).

Not until the 1960s and 1970s, after Mandelbrot and his colleagues introduced "fractals" and "self-similar" processes, the theory of time series was not enough developed to address long-range dependence (in price series see [85, 91, 90]). A stationary process Y_t has the long memory property, if for its autocorrelation function $\rho(k) = Cov(Y_t, Y_{t+k})/Var(Y_t)$ it holds that

$$\sum_{k=-\infty}^{\infty} |\rho(k)| = \infty$$

A long memory series has an autocorrelation function that decays hyperbolically, much more slowly than the geometrically tail off by "short memory" (ARMA) processes. Thus, it may be predictable at long horizons. The detailed empirical analysis on long-range dependence will be presented in the next chapter.

Skewness and Kurtosis

An random variable X's third and fourth standardized moments are also commonly used to present its stylized facts if exist. Called skewness and kurtosis, they are defined by

$$Skewness = E\left[\left(\frac{X-\mu}{\sigma}\right)^3\right]$$

$$Kurtosis = E\left[\left(\frac{X-\mu}{\sigma}\right)^4\right]$$

where μ and σ are mean and standard deviation accordingly. For observations of X: x_1, x_2, ... x_n, the sample skewness and kurtosis are defined respectively as

$$S = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i-\overline{x})^3}{\hat{\sigma}^3},$$

and

$$K = \frac{\sum_{i=1}^{n}(x_i-\overline{x})^4}{n\hat{\sigma}^4}.$$

where we define $\hat{\sigma}^2 = \frac{\sum_{i=1}^{n}(x_i-\overline{x})^2}{n}$ and $\bar{x}$ denotes the sample mean.

Kurtosis gives a measure of the thickness in the tails of a probability density function. For a normal distribution the kurtosis is 3. We further define for X, the Excess kurtosis (E_K) is calculated as

$$E_K = K - 3$$

It follows that, for a normal distribution, the excess kurtosis is 0. A fat tail distribution has a value of kurtosis that exceeds 3. That is, excess kurtosis is positive.

In Appendix, Tables 6.2-6.4 give the skewness and kurtosis for commodity price increments, and Figure 6.1 and 6.2 also show the histogram plot for commodity data. One can see all the time series samples have skewed and kurtotic increments. Generally speaking, if data is kurtotic, the prices will crash more often than expected, if data is skewed, the prices will crash to the direction that the distribution is skewed [93].

In general, we observed that regardless of the type of the commodities, almost all daily index and future price time series have unit roots with station-

ary increments where also increments are kurtotic and skewed. We have also observed that most of the processes are persistent and they have long-range dependence structure.

Chapter 2

Generalized Hurst Exponent

2.1 Long Memory and R/S Hurst exponent

Modeling commodity prices has proven to be challenging. Since 1990, when financial data became widely available, the financial literature witnessed a surge of interest in analyzing LRD processes. For instance, Haubrich and Lo [54] analyzed the associations between the business cycle and long-term memory, while Tieslau [116] provided details of estimating long memory models to price and monetary series. Cont [26] studied the relevance of LRD processes to financial modeling and discussed basic principles of financial theory and possible economic explanations for their presence in financial time series.

From the last chapter we show most of the commodity prices appear to be non-stationary, long-range dependence (LRD), skewed and kurtotic with different fractional characteristics. More specifically, we are more interested in the long-range dependence characteristic. Popular indexes that measure the time series fractional characteristics are R/S and generalized Hurst exponents [89].

In this section we will describe the concept of Hurst exponent within R/S analysis framework. The R/S Hurst exponent was first proposed by the hydrologist Hurst to study and predict the Nile River floods [60]. In his study, Hurst discovered that the values of successive yearly run-offs show a certain level of dependency. This phenomenon could not be modeled using a process with independent increments so he developed a method that eventually became known today as the Hurst rescaled range analysis (R/S). First applications of Hurst exponent in fractal geometry were proposed in [90] and [87], as Hurst ex-

ponent is directly related to fractal dimension, and is a measure of a data series "mild" or "wild" randomness [100]. Fractal analysis has become more popular in the finance community recently in particular, one application is the use of Hurst exponent in financial or economical time series data. Peters [99] used the Hurst process and R/S Analysis in testing and researching Capital Markets. He introduced the Fractal Markets Hypothesis (FMH), which avoids the classical assumptions that returns are lognormal and uncorrelated for financial mathematical models.

The rescaled range, or, R/S analysis is able to distinguish a random series from a fractal series, irrespective of the distribution of the underlying series (Gaussian or non-Gaussian). Given a sequence of n observations time series $y_1, y_2, \cdots, y_T$, of full length T is divided into a number of shorter time series of length $n = T, T/2, T/4, ...$ The Hurst exponent, H, is defined in terms of the rescaled range as follows.

First find the mean:

$$\overline{y} = \frac{1}{n}\sum_{i=1}^{n} y_t.$$

Then R captures the maximum and minimum cumulative deviations of the observations y_t from its mean $\overline{y}$, and it is a function of time:

$$R(n) = \max_{1\leq t\leq n}\left[\sum_{i=1}^{t}(y_i - \overline{y})\right] - \min_{1\leq t\leq n}\left[\sum_{i=1}^{t}(y_i - \overline{y})\right]$$

$S(n)$ is the sample standard deviation of the original time series:

$$S(n) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \overline{y})^2}$$

Then the R/S ratio can be calculated. The Hurst exponent is denoted by H such that for some constant C, we have

$$\mathbb{E}(R/S)_n = Cn^H \quad as \;\; n \to \infty. \tag{2.1}$$

The Hurst exponent is referred to as the "index of long-range dependence", which provides a measure for long term memory and fractality of a time series. Due to its robustness, the Hurst exponent has broad applications in time series analysis with only a few underlying systems assumptions. The values of the

Hurst exponent range between 0 and 1. Based on the Hurst exponent value H, a time series can be classified into three categories.

(1) $H = 0.5$ indicates a random walk.

(2) $0 < H < 0.5$ indicates an anti-persistent (ergodic) series.

(3) $0.5 < H < 1$ indicates a persistent (trend) series.

The strength of mean-reverting increases as H approaches 0. A persistent series is trend reinforcing, which means the direction (up or down compared to the last value) of the next value is more likely the same as the current value. The strength of trend increases as H approaches 1. Most economic and financial time series are persistent with $H > 0.5$. A time series with a large Hurst exponent has strong trend, thus it's natural to believe that such time series are more predictable than those having a Hurst exponent close to 0.5.

Commodity prices are usually understood to be mean-reverting with R/S Hurst exponent smaller than $1/2$ (see [117] and the references therein). However, looking at a large set of daily future and index commodity prices, we observe that commodity price time series have unit roots with skewed, kurtotic and stationary increments whose R/S and generalized Hurst exponent (which will be defined in the next sub section) are usually greater than $1/2$. While the R/S Hurst exponent is related to the long-range dependence structure, the generalized Hurst exponent is related to the existence of the higher moments of a shock processes. As we will discuss later in this thesis, if R/S Hurst exponent of a time series is greater than $1/2$, it is a long-range dependent process.

Taking logarithm on both sides of the Hurst exponent equation (2.1), we have:

$$\log(\mathbb{E}(R/S)_n) = c + H\log(n) \quad as \;\; n \to \infty \tag{2.2}$$

Then we run a linear regression to estimate the Hurst exponent H. The slope of the regression line approximates the Hurst exponent.

In this thesis we compute the Hurst exponent of the set of daily index and future commodity prices based on equation 2.2. The results show that in all case except, wheat index, oats future and index the Hurst exponent is larger than $1/2$, which indicates the process is persistent. The results are reported in the Tables 6.2-6.4 in the Appendix.

Remark 2. It is interesting that the observation of $H > 1/2$ for almost all commodities. This does not only imply the persistence of the data but also shows

the consistency of our measurement as we obtained the same properties from product ACF and PACF plots. R/S is robust if H depends on the exponents characterizing long-range dependence and does not depend on the underlying distribution of the time series. An interesting implication of this fact is that even if we assume the shock processes are fat tail, one should not be worried about the existence of R/S Hurst exponent. Indeed the robustness of R/S holds if there is a number $d \in (0, 1/2)$ such that $H = d + 1/2$. The number d is a measure of the process's long-range dependence, and is roughly equal to the rate of the decay of c_i's in (2.4). We will discuss about long memory parameter d in detail later. For further discussion one can also see [108] and [4].

The model of an Autoregressive Fractionally Integrated Moving Average (ARFIMA) process has often been referred as defining a time series with long-range dependence, or long memory.

Definition 4. An ARFIMA (p, d, q) model is an ARMA model where the innovations are fractional white noise, which can be rewritten in operator notation as:

$$\phi(L)(1 - L)^d(Y_t - \mu) = \theta(L)\varepsilon_t$$

where d is the fractional integration parameter allowed to take non-integer values, $\phi(L) = (1 - \phi_1 L - \phi_2 L^2 - \cdots - \phi_p L^p)$ specifies the AR lag polynomial, and $\theta(L) = (1 + \theta_1 L + \cdots + \theta_q L^q)$ specifies the MA lag polynomial. When $|d| \neq 0.5$, the process can be considered as a long memory process.

Formally, ARFIMA process $\{y_t\}_{t=0,1,2,..}$ can be introduced as a process whose fractional increment $z'_t = (I - L)^d (z_t)$ is ARMA$(p - 1, q)$, where L is the lag operator and $\{z_t\}_{t=0,1,2,...}$ is the increments of the original data $\{y_t\}_{t=0,1,2,...}$.

By invertibility of the operator $(1 - L)^d$, $z'_t = (I - L)^d (z_t)$ implies $z_t = (1 - L)^{-d} (z'_t)$. The operator $(I - L)^{-d}$ can be introduced by using its Taylor expansion $(I - L)^{-d} = I + dL/1! + d(d + 1)L^2/2! + d(d + 1)(d + 2)L^3/3! + \cdots$ (i.e. $z_t = z'_t + dz'_{t-1} + d(d + 1)z'_{t-2}/2! + d(d + 1)(d + 2)z'_{t-3}/3! + \cdots$) needs to exist (see [57, 46]). It is shown that for $-\frac{1}{2} < d < \frac{1}{2}$ the increments of an ARFIMA$(p - 1, 1 + d, q)$ process is a stationary process with long memory (see [57, 46]).

Fractional ARIMA or ARFIMA model is proven to be a useful way to capture the long memory of a process. For instance, in [9, 10] the long-range dependence of commodity prices were investigated by using ARFIMA processes. Their observations of low frequency monthly spot (in [9]) and future (in [10]) prices

suggested that the commodity prices in most cases have long-range dependency and can be modeled by using non-linear ARFIMA processes. MLE approach was used to estimate ARFIMA models for the macroeconomic time series [31]. Baillie [6] also found estimated ARFIMA model provides a good explanation of the behavior of US CPI inflation.

In the following sections, we first study the ARFIMA(p, d, q) models with skewed and kurtotic shocks for modeling commodity daily index and future prices. However, since non-linear models are very difficult to implement in practice, we study linear models with fat tail shock processes. Even though linear models with fat tail shock processes cannot generate long memory time series, they can generate time series with generalized Hurst exponents similar to the ones from the commodity daily index and future prices. This suggests that with regards to the generalized Hurst exponent, linear ARIMA$(p, 1, q)$ models with $\alpha-$stable shock[1] processes are good models with statistical characteristics very similar to commodity daily prices.

2.2 Generalized Hurst Exponent for Time Series Model

Given that almost all our commodity data series have unit roots with stationary increments, and Hurst exponents mostly greater than1/2, we are motivated to study processes that can have, and potentially can be used to generate these properties. Indeed, we want to observe later the same number for linear models, and compare them with the ones from commodity data.

The basic R/S Hurst exponent is related to the expected size of price changes, as a function of the lag between observations, as measured by $E\left(|y_t - y_{t-k}|^2\right)$. For the generalized form of the coefficient, the exponent here is replaced by a more general term, denoted by r. In this paper, we introduce the generalized Hurst function as follows. For a number $r > 0$, observe from [89]

$$H_r(k,t) = \frac{\log\left(\frac{E|y_t - y_{t-k}|^r}{E|y_t - y_{t-1}|^r}\right)}{r\log(k)}. \tag{2.3}$$

If $r = 2$ we simply use the notation $H(k,t)$. We will see later that for the processes we used in this paper, $H_r(k,t)$is independent from t, therefore at this

[1]Indeed $d = H - \frac{1}{2}$, where H denotes the R/S Hurst exponent.

moment we can assume $H_r(k,t) = H_r(k)$.

As we will see later if the shock processes of an $ARIMA(p,1,q)$ process have finite variance then $\lim_{k\to\infty} \log\left(\frac{E|y_t - y_{t-k}|^2}{E|y_t - y_{t-1}|^2}\right) / \log(k) = 1$. Thus, for $r = 2$ the time series behavior is consistent with a standard random walk, but exhibits properties more like the characteristics of fractional Brownian motion for $r \neq 2$.

Remark 3. We will discuss later that according to corollary 2, except the wheat future, soybean oil future, oats future, cocoa future and index, aluminum future, gold index, palladium index, and lean hogs index, in all other cases the independent shock process cannot have finite variance, since otherwise we had to have generalized $H = 1/2$. This result reinforces our assumptions that these processes are more likely to experience extreme values.

2.3 Time Series with Finite and Infinite Variances Shocks

In this section we discuss how linear autoregressive processes with infinite variance shock processes have long memory, which can be used in replace of nonlinear autoregressive time series. Our research shows that the generalized Hurst exponent of an ARIMA process with α-stable shocks is asymptotically greater than 1/2. This is consistent with our observation from the commodity daily index and future prices. Not only this, α-stable shocks (defined below) can produce processes that generate skewed and kurtotic data, which better justifies the use of ARIMA processes with α-stable shocks.

Stationary time series have different representations. Among them, we are interested in the canonical representation of the stationary processes. Let $\{y_t\}_{t=0,1,2,\cdots}$ be a stationary process, then there exists $\{\epsilon_t\}_{t\in\mathbb{Z}}$, a sequence of uncorrelated and identical distributed random variables and a sequence of real numbers $\{c_i\}_{i=0,1,2,,\cdots}$ such that

$$y_t = \mu + \sum_{i=0}^{\infty} c_i \epsilon_{t-i}. \tag{2.4}$$

Before we introduce stationarity, we will discuss the concept of unit root first. In time series , unit root arises when either the autoregressive or moving average polynomial of an ARMA model has a root on or near the unit circle.

One straight forward application is that a root near 1 of the autoregressive polynomial indicates the differences need to be applied to the data set before fitting an ARMA model, whereas a root near 1 from the moving-average polynomial could be due to data over differencing [18].

Remark 4. $ARMA(p,q)$ time series, $y_t = \mu' + \sum_{i=1}^{p} a_i y_{t-i} + \sum_{j=0}^{q} \theta_j \epsilon_{t-j}$ has an unit root if and only if $\sum_{i=}^{p} a_i = 1$.

The theorem suggests that any sequence type ARMA(p,q), $y_t = \sum_{i=1}^{p} a_i y_{t-i} + \sum_{j=0}^{q} \theta_j \epsilon_{t-j}$, in which the lag coefficients $\sum_{i=}^{q} a_i \neq 1$ has a limit in probability and time of either infinite or zero. It is known that if ϕ_y has root $\lambda = 1$, the difference process $z_t := y_t - y_{t-1}$ is an $ARMA(p-1,q)$ process with the same shock process ϵ_t and

$$z_t = b_2 z_{t-1} - \cdots + b_p z_{t-p+1} + \sum_{j=0}^{q} \theta_j \epsilon_{t-j}.$$

It is well understood that a process $\{z_t\}_{t=0,1,2,...}$ is stationary if and only if all roots of θ_z are inside the unit root circle. In this study, we assumed z_t is stationary, therefore the roots $\lambda_2, ..., \lambda_m$ of are all inside the unite circle.

In that case, if the time series $\{y_t\}_{t=0,1,2,...}$ has finite variance, we have the following formula for the auto-correlation function:

$$\gamma_i = E\left[(y_{t+i} - \mu)(y_t - \mu)\right] = \sum_{l=0}^{\infty} c_{l+i} c_l.$$

Note that the auto-correlation only depends on lag i and not time t. However, one needs to be careful since the summation $y_t = \sum_{i=0}^{\infty} c_i \epsilon_{t-i}$ has to be meaningful in a "topology". For instance, if $\{\epsilon_i\}_{i\in\mathbb{Z}}$ belong to L^2 i.e., has finite variance, then $\sum_{i=0}^{\infty} c_i \epsilon_{t-i}$ is convergent in L^2 norm. Nevertheless, we can assume in an appropriate topology $y_t = \sum_{i=0}^{\infty} c_i \epsilon_{t-i}$ is convergent and we work with a time series whose representation can be given as an infinite summation (or infinite moving average, $MA(\infty)$).

2.3.1 α-stable Random Variables

Stable distributions are a class of probability distributions suitable for modeling heavy tails and skewness. Assume $\{\epsilon_i\}_{i\in\mathbb{Z}}$ is a random variable with character-

istic function equal to

$$\begin{aligned}\Phi_\epsilon(\tau) &= E\left[\exp\left(i\tau\epsilon\right)\right] \\ &= \begin{cases} exp(-|\tau|^\alpha[1 - I\beta tan\,\frac{\pi\alpha}{2}(sign\,\tau)]) & \alpha \neq 1 \\ exp(-|\tau|[1 + I\beta\frac{2}{\pi}(sign\,\tau)log\,|\tau|]) & \alpha = 1 \end{cases},\end{aligned} \tag{2.5}$$

with $\mu \in (-\infty, \infty)$ and $\mathcal{I} = \sqrt{-1}$. Then the random variable X is *stable* if and only if $X \overset{d}{=} \sigma\epsilon + \mu$, where $\sigma \neq 0$ and $\mu \in \mathbb{R}$.

We denote X is an α-stable random variable with mean μ, scaling parameter σ and skewness parameter β. For simplicity, we denote an α-stable random variable by $S_\alpha(\sigma, \beta, \mu)$. Note that if $\epsilon_0 \sim S_\alpha(1, \beta, 0)$ then $\epsilon = \sigma\epsilon_0 \sim S_\alpha(\sigma, \beta, 0)$. Later we will assume that the shock process $\{\epsilon_i\}_{i\in\mathbb{Z}}$ has a common distribution $S_\alpha(\sigma, \beta, 0)$. A simple application of this definition is to show that the summation of any sequence of α-stable random variables is α-stable. It is also easy to see that for any series of independent random variable in general, if $\epsilon_i \sim S_\alpha(\sigma_i, \beta, \mu_i), i \in \mathbb{N}$ we have that $\sum_{i=1}^\infty \epsilon_i \sim S_\alpha\left((\sum_{i=1}^\infty \sigma_i^\alpha)^{\frac{1}{\alpha}}, \beta, \sum_{i=1}^\infty \mu_i\right)$.

In the following, we discuss how linear autoregressive processes with infinite variance shock processes have long memory, which can be used in replace of non-linear autoregressive time series.

In [9, 10] the authors observed that the monthly spot and future commodity prices have long-range dependence property. That implies the ARFIMA model can be a good model to fit the data. Barkoulas et al. [9] also studied the long memory-ness of the commodity monthly data by estimating the fractal parameter d. Based on these existing studies we can see the nonlinear ARFIMA models are usually applied for finite variance shock processes.

However, our results about the generalized Hurst exponent show that the shock processes are unlikely to be thin tail and symmetric (see next sections). Recall that one can consider a representation in the form (2.4) for the increments of an ARFIMA$(p-1, 1+d, q)$ process, where $-1/2 < d < 1/2$. It is shown in [46] that up to a slow varying function[2], we must have $c_i \sim i^{d-1}$. This inspires us to consider a similar measure for the long-range memory with no finite variance, that is, the rate of decay of c_i. According to [4], if $d \in (0, 1 - \frac{1}{\alpha})$ for some $\alpha \in (1, 2)$, such that up to a slow varying function we have $c_i \sim i^{d-1}$, then for α-stable shock process, $H = d + 1/2$. Since for almost all commodities we observe that $H > 1/2$, we can see that $d > 0$, and hence, the process display

[2] A function L is slow varying if $\forall a > 0, \lim_{x\to\infty} \frac{L(ax)}{L(x)} = 1$.

the long-range memory.

Looking at the canonical representation (2.4) of the stationary processes, one can observe that in order to generate time series with skewed and kurtotic increments one needs to consider skewed and kurtotic shock processes. Therefore, in short, our observations suggest that a good model for commodity daily index and future prices is an ARFIMA(p, d, q) process with α-stable shocks. However, since non-linear autoregressive models are practically difficult to implement, in the next sections, we study another measure of fractionally of processes - generalized Hurst exponent, and show that if we focus our attention to this measure rather than the R/S Hurst exponent, linear models still can be good candidates for modeling actual data.

2.3.2 Generalized Hurst Exponent for ARIMA models

To study the generalized Hurst exponents of ARIMA processes, we need to explain few things about all ARMA processes first.

A standard way of studying an ARMA(p, q) process z_t is to represent it in the form of a vector AR(1) process. To study this problem we rewrite it in a vector form as follows

$$\underbrace{\begin{bmatrix} z_t \\ z_{t-1} \\ z_{t-2} \\ \vdots \\ z_{t-p+2} \end{bmatrix}}_{Z_t} = \underbrace{\begin{bmatrix} b_2 & b_3 & \cdots & b_{p-1} & b_p \\ 1 & 0 & \cdots & & 0 \\ 0 & 1 & \cdots & & 0 \\ \vdots & \vdots & \ddots & & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix}}_{B} \underbrace{\begin{bmatrix} z_{t-1} \\ z_{t-2} \\ z_{t-3} \\ \vdots \\ z_{t-p+1} \end{bmatrix}}_{Z_{t-1}} + \underbrace{\begin{bmatrix} \sum_{j=0}^{q} \theta_j \epsilon_{t-j} \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}}_{\overrightarrow{\epsilon_t}}.$$

We can rewrite the ARMA(p, q) process as follows

$$Z_t = BZ_{t-1} + \overrightarrow{\epsilon_t}. \tag{2.6}$$

Using (2.6), we can represent the process Z_t as an infinite moving average process. In order to do that, we have to consider the i.i.d shock process $\{\epsilon_t\}_{t\in\mathbb{Z}}$, then iteratively (2.6) we get

$$Z_t = \sum_{l=0}^{\infty} B^l \overrightarrow{\epsilon_{t-l}}. \tag{2.7}$$

Denoting component $(1,1)$ of B^l by $\left[B^l\right]_{1,1}$ we get

$$z_t = \sum_{l=0}^{\infty} \left[B^l\right]_{1,1} \sum_{j=0}^{q} \theta_j \epsilon_{t-l-j}. \tag{2.8}$$

This means in the representation (2.4) i.e., $z_t = \sum_{l=0}^{\infty} c_l \epsilon_{t-l}$ of a stationary process, for $l > q$ we have $c_l = \sum_{j=0}^{q} \theta_j \left[B^{l-j}\right]_{1,1}$. It is well-understood that for any number $1 > \rho > \max\{|\lambda_2|, ..., |\lambda_m|\}$ where λ are roots of $\phi(L)$, there is an $M_\rho > 0$ such that

$$|c_j| \leq M_\rho \rho^j, j = 1, 2, ... \tag{2.9}$$

Note that if we have an $AR(p)$ process, meaning that $q = 0$, then we have $c_l = \left[B^l\right]_{1,1}$.

Remark 5. Inequality (2.9) shows that ARMA models can only generate processes with short-range dependence. However, since they are linear they can be used much easier for fitting and forecasting, and that is why we choose them for modeling commodity prices.

2.3.3 Generalized Hurst Exponent with Finite Variance Shocks

Let us assume that $\{y_t\}_{t=0,1,2,...}$ is a process with stationary increments. The variance of the one time-step increment can be expressed as

$$\begin{aligned} y_t - y_{t-k} &= (y_t - y_{t-k}) + (y_{t-1} - y_{t-1}) + \cdots + (y_{t-k+1} - y_{t-k+1}) \\ &= (y_t - y_{t-1}) + (y_{t-1} - y_{t-2}) + \cdots + (y_{t-k+1} - y_{t-k}) \\ &= z_t + z_{t-1} + \cdots + z_{t-k+1}. \end{aligned}$$

We have

$$\begin{aligned} E\left|y_{t+k}-y_t\right|^2 &= \sum_{i=t-k+1}^{t}\sum_{j=t-k+1}^{t} cov\left(z_i,z_j\right) \\ &= \sum_{i=t-k+1}^{t}\sum_{j=t-k+1}^{t}\gamma_{|i-j|} \\ &= \sum_{i=0}^{k-1}\sum_{j=0}^{k-1}\gamma_{|i-j|} \\ &= 2\sum_{i=1}^{k-1}\left(k-i\right)\gamma_i + k\gamma_0 \\ &= 2\sum_{j=1}^{k-1}\sum_{i=1}^{j}\gamma_i + k\gamma_0 \end{aligned} \tag{2.10}$$

where $\gamma_{|t_2-t_1|}=\gamma\left(|t_2-t_1|\right)=E\left(z_{t_1}z_{t_2}\right)$. First, observe that $E\left|y_{t+k}-y_t\right|^2$ is only a function of k, proving that $H\left(k,t\right)=H\left(k\right)$. We show as k goes to ∞, $H\left(k\right)$ converges to $1/2$.

Theorem 1. *Suppose* $\{y_t\}_{t=0,12,...}$ *is a process with stationary increments and* $\forall i\geq 1, \mathrm{Var}\left(\epsilon_i\right)<\infty$. *Then* $E\left|y_{t+k}-y_t\right|^2\sim O\left(k\right)$ *iff* $\lim_{j\to\infty}\sum_{i=1}^{j}\gamma_i$ *is convergent.*

Proof. By using the Cesaro summability theorem, one gets from (2.10)

$$\begin{aligned} \lim_k \frac{E\left|y_{t+k}-y_t\right|^2}{k} &= \lim_k 2\sum_{j=1}^{k-1}\frac{\sum_{i=1}^{j}\gamma_i}{k}+\gamma_0 \\ &= 2\sum_{i=1}^{\infty}\gamma_i+\gamma_0. \end{aligned}$$

□

Corollary 1. *With all assumptions of the previous theorem,* $\lim_{k\to\infty}H\left(k\right)\leq\frac{1}{2}$ *iff* $\lim_{j\to\infty}\sum_{i=1}^{j}\gamma_i$ *is convergent.*

An implication from (2.4) and (2.9) is that if shocks have finite variance, for $1>\rho>\max\left\{|\lambda_2|,...,|\lambda_m|\right\}$, there exists K_ρ such that

$$E\left(z_{t+i}z_t\right)\leq K_\rho\rho^i,\ i\geq 1,$$

or

$$|\gamma_i| \leq K_\rho \rho^i,\ i \geq 1. \tag{2.11}$$

Using this last inequality, for a large number M we have

$$\begin{aligned} \left| \sum_{j=M}^{\infty} \gamma_j \right| &\leq \sum_{j=M}^{\infty} K\rho^j \\ &\leq \frac{K_\rho \rho^M}{1-\rho}. \end{aligned} \tag{2.12}$$

This implies that $\lim_{M\to\infty} \sum_{j=1}^{M} \gamma_j$ exists and is finite. We denote this limit by γ.

As one can see, when the shocks have finite variance, the rate of the covariance process decays exponentially, therefore the process has short memory. Now by using Corollary 1 we have

Corollary 2. *For an ARIMA$(p, 1, q)$ process with finite variance shocks we have*

$$\lim_{k\to\infty} H(k) \leq \frac{1}{2}.$$

This corollary confirms that the generalized Hurst exponent of an ARIMA process with finite variance shocks is asymptotically 1/2. This is not consistent with our observation from the commodity daily index and future prices, which in almost all cases (except the wheat future, soybean oil future, oats future, cocoa future and index, aluminum future, gold index, palladium index, and lean hogs index) generalized $H > 1/2$. In the next section we will see that the result is different for the fat tail shock process.

2.3.4 Generalized Hurst Exponent with $\alpha-$stable Shocks

In the previous sections we observe that the commodity daily index and future prices are skewed, kurtotic and have generalized Hurst exponent larger than 1/2. This motivates us to study ARIMA processes with skewed and kurtotic shocks. Among all such shock processes, we find the α-stable shock processes more convenient to work with[3]. First, they can be skewed and leptokurtic (as opposed to normal shocks), and also (as we will see) can yield processes

[3]It is very likely that the same is true for the shock process whose tail behaves similar to the tail of an α-stable shock i.e., $F_\epsilon(x) \sim 1 - 1/x^\alpha, x \to \infty$. However, since studying the general case is beyond the scope of this paper we leave it to interested reader.

with generalized Hurst exponents larger than 1/2. On the other hand, the self similarity of α-stable processes makes it very convenient to find the Hurst function of an ARIMA process α-stable shocks.

As discussed for any stationary ARMA(p,q) process z_t there exists a sequence $\{c_i\}_{i=1}^{\infty}$ and an i.i.d sequence $\{\epsilon_i\}_{i\in\mathbb{Z}}$ such that

$$z_t = \sum_{i=0}^{\infty} c_i \epsilon_{t-i}.$$

Now let us assume that $\epsilon_i \sim S_\alpha(\sigma,\beta,0), i \in \mathbb{Z}$ and also $\sum_{i=0}^{\infty} |c_i|^\alpha < \infty$.

For our ARIMA$(p,1,q)$ process y_t with increments z_t, we have that

$$\begin{aligned} y_t - y_{t-k} &= (y_t - y_{t-1}) + ... + (y_{t-k+1} - y_{t-k}) \\ &= z_t + \cdots + z_{t-k+1} = \sum_{j=0}^{k-1}\sum_{i=0}^{\infty} c_i \epsilon_{t-j-i}. \end{aligned} \tag{2.13}$$

To visualize this we look at the following development

$$\begin{array}{lllllll} y_t - y_{t-k} = & c_0\epsilon_t + & c_1\epsilon_{t-1} + & c_2\epsilon_{t-2} & \cdots & c_{k-1}\epsilon_{t-k+1} & +c_k\epsilon_{t-k} \quad \cdots \\ & & +c_0\epsilon_{t-1} + & c_1\epsilon_{t-2} & \cdots & & \\ & & & +c_0\epsilon_{t-2} & \cdots & & \\ & & & & \ddots & & \\ & & & & & +c_0\epsilon_{t-k+1} & c_1\epsilon_{t-k} \quad \cdots \end{array} .$$

If we let $c_i = 0$ for $i = -1, -2, \cdots$, we get that $y_t - y_{t-k} = \sum_{i=0}^{\infty}\left(\sum_{j=0}^{k-1} c_{j+i-k+1}\right)\epsilon_{t-i}$. Therefore,

$$y_t - y_{t-k} \sim S_\alpha\left(\left(\sum_{i=0}^{\infty}\left|\sum_{j=0}^{k-1} c_{j+i-k+1}\right|^\alpha\right)^{\frac{1}{\alpha}}, \beta, 0\right) \tag{2.14}$$

Let $0 < \alpha < 2$. It is known that for any α-stable random variable $\epsilon \sim S_\alpha(\sigma,\beta,\mu)$ and for every $0 < r < \alpha < 2$, r-th moments are fine. However, if $\alpha = 2$, then for any $r \geq 0$, r-th moments are fine. Let $\epsilon_0 \sim S_\alpha(1,\beta,0)$, then by (2.14) we get

$$E\left|y_t - y_{t-k}\right|^r = E\left(\left|\left(\sum_{i=0}^{\infty}\left|\sum_{j=0}^{k-1} c_{j+i-k+1}\right|^{\alpha}\right)^{\frac{1}{\alpha}}\epsilon_0\right|^r\right)$$
$$= \left(\sum_{i=0}^{\infty}\left|\sum_{j=0}^{k-1} c_{j+i-k+1}\right|^{\alpha}\right)^{\frac{r}{\alpha}} E\left(\left|\epsilon_0\right|^r\right).$$

This implies

$$\log\left(\frac{E\left|y_t - y_{t-k}\right|^r}{E\left|y_t - y_{t-1}\right|^r}\right) = \log\left(\frac{\left(\sum_{i=0}^{\infty}\left|\sum_{j=0}^{k-1} c_{j+i-k+1}\right|^{\alpha}\right)^{\frac{r}{\alpha}} E\left(\left|\epsilon_0\right|^r\right)}{\left(\sum_{i=0}^{\infty}\left|c_{i-k+1}\right|^{\alpha}\right)^{\frac{r}{\alpha}} E\left(\left|\epsilon_0\right|^r\right)}\right)$$
$$= \log\left(\left(\frac{\sum_{i=0}^{\infty}\left|\sum_{j=0}^{k-1} c_{j+i-k+1}\right|^{\alpha}}{\sum_{i=0}^{\infty}\left|c_{i-k+1}\right|^{\alpha}}\right)^{\frac{r}{\alpha}}\right) \tag{2.15}$$
$$= \frac{r}{\alpha}\log\left(\frac{\sum_{i=0}^{\infty}\left|\sum_{j=0}^{k-1} c_{j+i-k+1}\right|^{\alpha}}{\sum_{i=0}^{\infty}\left|c_{i-k+1}\right|^{\alpha}}\right)$$

Corollary 3. *Let $0 < \alpha \leq 2$ and $r > 0$. Furthermore, if $0 < \alpha < 2$, we assume $0 < r_1 < r_2 < \alpha$. Then*

$$H_{r_1}(k) = H_{r_2}(k).$$

Proof. Given (2.15) we have

$$H_r(k) = \frac{\log\left(\frac{E\left|y_t - y_{t-k}\right|^r}{E\left|y_t - y_{t-1}\right|^r}\right)}{r\log(k)} = \frac{\frac{1}{\alpha}\log\left(\frac{\sum_{i=0}^{\infty}\left|\sum_{j=0}^{k-1} c_{j+i-k+1}\right|^{\alpha}}{\sum_{i=0}^{\infty}\left|c_{i-k+1}\right|^{\alpha}}\right)}{\log(k)}.$$

□

By the previous corollary, the following definition of H_α is indeed consistent with the original definition (2.3).

$$H_\alpha(k,t) = H_\alpha(k) = \frac{\log\left(\frac{\sum_{i=0}^{\infty}\left|\sum_{j=0}^{k-1} c_{j+i-k+1}\right|^{\alpha}}{\sum_{i=0}^{\infty}\left|c_i\right|^{\alpha}}\right)}{\alpha\log(k)}.$$

Corollary 4. *Let* $1 < \alpha \leq 2$ *and assume that shock sequence* $\{\epsilon_i\}_{i\in\mathbb{Z}}$ *is i.i.d and* α*-stable. Then*

$$\lim_{k\to\infty} \frac{\log\left(\frac{\sum_{i=0}^{\infty}\left|\sum_{j=0}^{k-1} c_{j+i-k+1}\right|^{\alpha}}{\sum_{i=0}^{\infty}|c_{i-k+1}|^{\alpha}}\right)}{\log(k)} = 1.$$

Proof. Proof in the Appendix. □

Theorem 2. *Suppose* $0 < \alpha < 2$ *and* $0 < r < \alpha$*, or* $\alpha = 2$ *and* $r > 0$*. Then*

$$\lim_{k\to\infty} H_r(k) = \frac{1}{\alpha}.$$

This theorem confirms that the generalized Hurst exponent of an ARIMA process with α-stable shocks is asymptotically greater than 1/2. This is consistent with our observation from the commodity daily index and future prices. Not only this but α-stable shocks can produce processes that generate skewed and kurtotic data, which better justifies the use of ARIMA processes with α-stable shocks.

2.4 Simulation and Testing

In this section, we are going to examine the robustness of the generalized Hurst exponent estimation to i.i.d. non-normal process with heavy tails. As introduced in the second section, α-stable random variable follows distribution with parameters mean μ, $\mu \in \mathbb{R}$, scaling parameter σ, $\sigma > 0$, and skewness parameter β, $-1 \leq \beta \leq 1$. For simplicity, we apply $\mu = 0$, $\sigma = 1$ and $\beta = 0$ symmetric distribution. Here we use i.i.d. $\alpha-$stable distributed random variables from $S_\alpha(1, 0, 0)$ with different value of parameter $\alpha \in (0, 2]$ in the stable distributions, to simulate 100 time series with lengths of 5000.

It is interesting to see how the generalized Hurst exponent changes with the different α for various lags lengths of time series. In other words, we simulate the grid of independent identically distributed stable increments with varying coefficient α and k. In this case, we have $0.1 \leq \alpha \leq 2$ with a step of 0.1, and $30 \leq k \leq 350$ with a step of 10. For each position in the grid, we then estimate generalized Hurst exponents. The results of Table 6.6 and 6.7 in the Appendix summarize all the expected value of a whole grid of the simulated results, showing how the generalized Hurst exponent changes for different α

values. Also we can observe that the generalized Hurst exponent is converging to $1/\alpha$ when lags k goes to the maximum number of trading days (here we assume 350 to be the number of trading days per year).

The figure 2.1 also display the results of generalized Hurst exponents from 100 time series simulations plotted in the gray dashed lines, with the heaviest tails ($\alpha = 0.1$) to the normal distribution ($\alpha = 2$), while the red line represents the mean value. The X-axis goes from the simulation with the heaviest tails to the simulations with the normal distribution. It is clearly shown that expected value of the generalized Hurst exponents is slowly converging to 0.5 with α changing from 0 to 2.

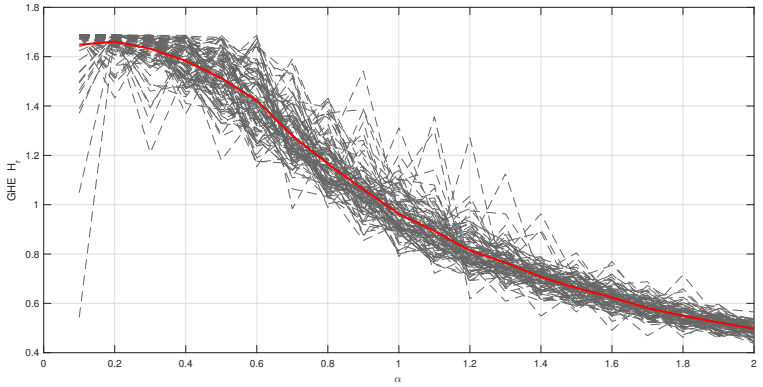

Figure 2.1: Generalized Hurst Exponents Estimation.

2.5 Investigation of Commodity Daily Index and Futures Prices

Recall, we have also observed that almost all daily index and future price time series have unit roots with stationary increments where also increments are kurtotic and skewed. In addition, we observe that, except wheat index, oats index and future prices, daily commodity prices have R/S Hurst exponent greater than $1/2$. This shows that almost all the processes are persistent and they have long-range dependence structure. Based on existing literature as well as our observations, ARFIMA(p, d, q) process with α-stable shocks can be a good model for modeling commodity prices.

We have also observed that except for the wheat future, soybean oil future, oats future, cocoa future and index, aluminum future, gold index, palladium index, and lean hogs index, all other commodities have generalized Hurst exponent is greater than 1/2. The results are reported in the Table 6.5 in the Appendix. Since general Hurst exponent is used to measure the long memory of the underlying data sets, the traditional bootstrap method cannot be applied due to the lack of replication features for long-range dependence.

Table 6.8 shows the log-likelihood for all the commodities fitted with ARIMA models with α-stable residuals and normal distributed residuals together with estimated values of α. One can see the log-likelihoods are relatively close indicating similar model fitting results and modeling abilities. Further research may be needed on developing some goodness-of-fit test to see if one model is better than the other.

We still do not have enough explanation for the fractionality and variety of R/S and generalized Hurts exponent of commodities. However, as it was discussed in [26], one can name few reasons. For instance, heterogeneity in time horizons of economic agents, where long-term investors naturally focus on long-term behavior of prices and traders aim to exploit short-term fluctuations [75, 50], by using genetic algorithms for decision rules [3] and switching between trading strategies. Switching of economic agents between two behavioral patterns leads to large, aggregate fluctuations in the financial market [82, 71] and finally investor inertia, where the duration of regimes that agents holding their strategies seems to be more important [78, 11]. In addition, Samorodnitsky [108], Doukhan et al. [36] attempted to explain the long-range dependence in other financial time series data where the dependence properties of financial time series are discussed through the empirical behavior of auto-correlation functions. *GARCH* model [14, 13] and LRD were used to take into account the volatility clustering phenomenon (see [27, 35, 50, 13]). Since the LRD are often formulated in terms of self-similar process [12], self-similarity does not imply LRD in any way and self-similar processes may originate from different process like fractional Brownian motions, α-stable Lévy process, etc. This is why we also study $\alpha-$stable shocks in our analysis . Cont [26] argued that theoretical restrictions imposed by arbitrage are quite weak and cannot be used as arguments to exclude a family of stochastic processes as potential models, and it is more interesting to use fractional processes as models of volatility (see [98, 7, 24]).

Chapter 3

Multidimensional Data Clustering

3.1 Cluster Analysis and Feature Learning

Cluster analysis is a very important analysis and is often required in real-world problems. It is a statistical tool used for grouping a set of objects so that objects in a same group, or a cluster, are more closely related to each other than to objects in other groups or clusters. This can identify mutually homogeneous groups (clusters) of observations such that the within-group-object similarity is maximal and the between-group-object similarity is minimal of some random variable X. An object can be described by a set of measurements, or by its relation to other objects. Assuming the observations in the same cluster share similar statistical properties, that is, have a highly correlated inner relationship, this will provide us with more useful information for modeling and forecast future scenarios.

It is well-known that the measured commodity data observations are usually high-dimensional and contain significant redundant information. A large number of features can cause learning models to overfit and affect their performance. To address this challenge, many existing approaches apply dimensionality reduction on high-dimensional data either based on feature extraction or feature selection before standard clustering modeling. In machine learning, dimensionality often refers to the number of features or input variables in a dataset. Recently, Principal Component Analysis (PCA) is commonly used for feature

extraction which creates low-dimensional representations of the original feature set. Feature selection finds an appropriate subset of the original features in order to filter irrelevant or redundant variables. Dimensionality reduction technique is often able to efficiently obtain features from very high dimensional datasets that works well in practice, however, discriminant information that is hidden in different subspaces can be lost when many irrelevant dimensions are filtered.

Clustering algorithms divide, or partition, data into natural groups of objects. By natural it means that the objects in a cluster should be internally similar to each other, but differ significantly from the objects in the other clusters. There are many clustering methods in literature, ranging from nonparametric approaches in defining specific distances or dissimilarities for functional data, such as variants of the k-means method [114] and clustering after transformation and smoothing [110] to k-centers functional clustering approach [23]. Most recently, the model-based procedure for clustering functional data has become popular, such as curves clustering method using functional random variables density approximation [61, 62] and expend coefficients of the curves into a spline basis of functions that distribute according to a mixture Gaussian distribution [63].

When consider the modeling of high-dimensional data, one can categorize them into groups based on the model characteristics of individual series or the residuals after model fitting. Such properties are called the functional nature of the data [77]. Model-based functional data clustering takes the functional nature of the data. In this section we study two different model-based functional data clustering methods based on both time domain and frequency domain. The first approach has been developed in analysing functional data on time domain to approximate density of functional random variables. Functional observations are serially correlated over time, where each curve represents a segment of the whole time interval. The second functional data clustering on the frequency domain can simultaneously estimate Spectral Density Functions (SDFs) for a collection of stationary time series that share some common features. The main purpose of this section is to efficiently distinguish different time series, choose the optimal structure of data set so that we can achieve the most efficient data features.

3.2 Model-based Functional Data Clustering in Time Domain

One of the effective methods to explore the relationships among data and clusters is the Functional Data Clustering. Such a method enables us to detect patterns and find clusters with high-dimensional time-dependent functional data. When study the clusters, particularly when the underlying data is observed frequently in time domain, Functional Data Analysis (FDA) is commonly used for multiple time series.

In this section, we first outline the techniques of functional data analysis and functional principal components analysis to transform discrete observations into an optimal representation of curves into a function space of reduced dimension. Then, the approximation of the density function for multivariate functional data is introduced.

3.2.1 Modeling Functional Data

In Functional data analysis (FDA), functional data are typically observed discretely and each time series is treated as observations of a continuous function collected at a finite time points. In theory, however, observations are assumed to be in an infinite dimensional space, for example, some random variables X taking values from an infinite dimensional space, such as a space of functions defined on some continuous set T.

According to [39], a functional random variable X is a random variable with values in an infinite dimensional space[1]. In order for the clustering algorithms to perform on finite dimensional data, discretely observed data need to be transformed into continuous functions or curves (functional data). To do so, the most popular approach is to reduce the infinite dimensional problem to a finite one by approximating data with elements from some finite dimensional space.

From [39], the important features of functional data are 1) functional data are infinite dimensional; 2) the measurements within one curve display high correlation.

Definition 5. Let X be a vector space over $\mathbb{C}$. An inner product on X is a map $(\cdot,\cdot) : X \times X \to \mathbb{C}$ such that, for all $x, y, z \in X$ and $\lambda \in \mathbb{C}$:

1. $(x, y) = (y, x)$ (Symmetry)

[1]The random variable underlying data is a stochastic process $X = \{X(t), t \in T\}$

2. $(x + y, z) = (x, z) + (y, z)$ (Bilinearity)

3. $(\lambda x, y) = \lambda(x, y)$ (Homogeneity)

4. $(x, x) \geq 0$ with equality iff $x = 0$ (Positivity)

Definition 6. A complete inner-product space is called a Hilbert space H.

Consider $X = \{X_t\}_{t \in T}$ taking values in a Hilbert space H of functions defined on time set T, the underlying model for X_i's is generally an i.i.d. sample of random variables with the same distribution as X.

Define a basis $\Phi = \{\phi_1, \phi_2, \cdots, \phi_L\}$ generating some space of functions in H, X can have the basis expansion for some $L \in \mathbb{N}$, $\alpha_{il} \in \mathbb{R}$.

$$X_i(t) = \sum_{l=1}^{L} \alpha_{il} \phi_l(t)$$

Then, the sample paths basis coefficients can be estimated from discrete-time observations.

3.2.2 Functional Principle Components Analysis (FPCA)

Principal Component Analysis (PCA) is a very popular dimensionality reduction technique that captures the maximum information presented in the original data and simultaneously minimizes the difference between the original data and the new reduced dimensional representation [64, 103, 102].

PCA is widely applied to reduce the number of variables in a model as well. Finding a small number of linear combinations of the variables that account for most of the variance in the observed data is the most common approach that motivates the PCA [58], obtaining linearly uncorrelated variables stemming from linearly correlated variables.

One way to operate PCA is by solving the characteristic polynomials of a system. Denote covariance matrix as $\mathbf{C}$,which is symmetric and can be rewritten as $\mathbf{C} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^{-1} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^{T} = \sum_{i=1}^{n} \lambda_i \mathbf{q}_i \mathbf{q}_i^T$, where matrix $\mathbf{Q}$ consists of eigenvectors $\mathbf{q}_i$ of $\mathbf{C}$ and $\mathbf{\Lambda}$ is a diagonal matrix with eigenvalues λ_i of $\mathbf{C}$. The characteristic polynomial of C is given by $det(\mathbf{C} - \lambda\mathbf{I}) = 0$, where $det(\cdot)$ is the determinant and $\mathbf{I}$ is the identity matrix. By solving the polynomial, one can get the eigenvalues λ_i and corresponding eigenvector $\mathbf{q}_i$. In PCA the eigenvalues are sorted in descending order, i.e. $\lambda_i \geq \lambda_{i+1}$, and as such the first corresponding eigenvector has the highest degree of explanation of the eigenvectors in $\mathbf{Q}$.

Another approach of PCA is to obtain best low-rank matrix approximation of the data matrix [64]. The two approaches compute the spectral decomposition of the sample covariance matrix and the singular value decomposition (SVD) of the data matrix respectively, and give the same result.

Different from the traditional functional basis expansion, where pre-specified sets of basis functions are needed, Functional Principal Component Analysis (FPCA) is a key technique in FDA due to its leading role in extracting the features and characterizing a set of curves. Thus, when dealing with discrete data, FPCA is carried out as a preprocessing step to find and reconstruct the functional form of data.

FPCA generalizes the Functional Principal Components (FPCs) through data-adaptive basis functions that are determined by the covariance function of functional data from $\mathbb{R}^n$ to the L^2 space. It is determined by principal components characterizing the variability of density via the eigenfunctions corresponding to the ranked eigenvalue of an empirical covariance operator.

In short, given functional data $\{X_1, \ldots, X_n\}$, FPCA is a tool that gives us an optimal representation of curves into a function space of reduced dimension from $\mathbb{R}^n$ to the L^2 space when clustering functional data, by determining principal components via the eigenfunctions corresponding to the largest, second largest, etc. eigenvalue of an empirical covariance operator. Specifically, dimension reduction is achieved through an expansion of the underlying $X_i(t)$ in a functional basis that consists of the eigenfunctions the covariance operator of the process X.

Consider functional random variable X as a L_2-continuous stochastic process[2], i.e.:

$$\forall t \in \mathrm{T},\ \lim_{h \to 0} E[|X(t+h) - X(t)|^2] = 0$$

Let $\mu = \{\mu(t) = E[X(t)]\}_{t \in T}$ denotes the mean function X. The covariance operator γ of X is an integral operator:

$$\begin{aligned} \gamma : L_2(T) &\longrightarrow L_2(T) \\ f &\xrightarrow{\gamma} \gamma f = \int_0^T k(\cdot, t) f(t) dt \end{aligned} \tag{3.1}$$

with kernel k defined by the covariance function:

[2]Under mild assumptions, the underlying stochastic process can be expressed as a countable sequence of uncorrelated random variables FPCs, which in many practical applications is truncated to a finite vector.

$$k(s,t) = E[(X(s) - \mu(s))(X(t) - \mu(t))], \quad s,t \in T$$

Under the L_2-continuity hypothesis, the mean and the covariance function are continuous and the covariance operator γ is a Hilbert-Schmidt one.

Definition 7. *Hilbert-Schmidt Operator*

Let $D \subset \mathbb{R}^n$ be a bounded domain. A function $k : D \times D \to \mathbb{R}$ is called a Hilbert-Schmidt kernel if $\int_D \int_D |k(x,y)|^2 dxdy < \infty$. For every $u \in L^2(D)$ the integral operator $K : L^2(D) \to L^2(D)$ given by

$$Ku(x) = \int_D k(x,y)u(y)dy$$

is compact, positive and self-adjoint.

In the case of a compact, positive and self-adjoint operator, the eigenvalues are non-negative. The spectral analysis of γ provides a countable set of positive eigenvalues $\{\lambda_j\}_{j\geq 1}$ associated to an orthonormal basis of eigenfunction $\{\psi_j\}_{j\geq 1}$:

$$\gamma\psi_j = \lambda_j\psi_j$$

with $\lambda_1 \geq \lambda_2 \geq \cdots \geq 0$.

Since $\gamma\psi_j = \int_D k(s,t)\psi_j(s)ds$ from eq(3.1), then we have:

$$\int_0^T k(s,t)\psi_j(s)ds = \lambda_j\psi_j(t), \quad s,t \in T$$

Karhunen and Loève [67, 81] independently discovered the FPCA expansion, also known as Karhunen-Loève expansion. The Karhunen-Loève expansion of the curves allows each cluster's parameters to be of different sizes, based on the quantity of variance expressed by the corresponding FPCA.

Definition 8. Karhunen-Loève Expansion

$$X(t) = \mu(t) + \sum_{j=1}^{\infty} C_j\psi_j(t) \ , \quad t \in T \tag{3.2}$$

where μ is the mean function of X, $\{C_j\}_{j\geq 1}$ are the functional principal com-

ponents (FPCs) of X, sometimes referred to as scores:

$$C_j = \int_0^T (X(t) - \mu(t))\psi_j(t)dt,\ j \geq 1$$

and ψ_j form an orthonormal system of eigenfunctions of the covariance operator of X:

$$\int_0^T k(s,t)\psi_j(s)ds = \lambda_j\psi_j(t), \quad s,t \in T$$

where $\{C_j\}_{j\geq 1}$ are zero-mean uncorrelated random variables with variance λ_j.

3.2.3 Approximation of the Density for Multivariate Functional Data

Consider the principal components indexed by the descending order of the eigenvalues ($\lambda_1 \geq \lambda_2 \geq \cdots$). Let $X^{(q)}$ denotes the approximation of X by truncating eq(3.2) at the first q terms, $q \geq 1$

$$X^{(q)}(t) = \mu(t) + \sum_{j=1}^{q} C_j\psi_j(t)\ , \quad t \in T \tag{3.3}$$

Hence, $X^{(q)}$ is the best approximation of X under the mean square criterion [33]. Denoting by $||.||$ the usual norm on $L_2(T)$, the convergence of the sum in eq (3.3) holds such that $E(||X - X^{(q)}||^2) = \sum_{j\geq q+1} \lambda_j$ and $||X - X^{(q)}|| \overset{q\to\infty}{\longrightarrow} 0$.

Similar dimension reduction can be achieved by expanding the functional data into other function bases, such as spline, Fourier, or wavelet bases. The particular feature of FPCA is it uses principal components for a fixed q, so the FPC expansion explains most of the variation in X in L^2. When choosing q in an estimation setting, there is a trade-off between bias and variance. Thus, model selection procedure is needed to obtain consistency of the representation.

Model-based clustering based on mixture models has been widely used in clustering multivariate data and has been extended to functional data clustering. For example, Jacques and Preda [61] applied functional clustering models based on Gaussian Mixture Models, assuming each class is represented by a Gaussian probability density, to identify homogeneous groups of data sampled from a mixture densities model.

Recall eq(3.3), assume X is a Gaussian process, and the principal components C_j are Gaussian and independent. Then the density $f_X^{(q)}$ can be used as an approximation of the density of X with the following form:

$$f_X^{(q)}(x) = \prod_{j=1}^{q} f_{C_j}(c_j(x)) \tag{3.4}$$

where f_{C_j} is the principal component density and $c_j(x) =< x, \psi_j >_{L^2}$ is the jth component score of x.

Given functional data $\{X_1, \ldots, X_n\}$, consider when there exists a latent group variable Z, of K groups, such that $Z = Z_1, \ldots, Z_K$ with

$$Z_g = \begin{cases} 1 & \text{if X belongs to cluster g} \\ 0 & \text{otherwise} \end{cases} \qquad 1 \leq g \leq K$$

For each $i = 1, \ldots, n$ associated to X_i, the corresponding categorical variable $Z_{i,g}$ indicates the cluster group X_i belongs. Then each couple $(X_i, Z_{i,g})$ is assumed to be an independent realization of the random vector (X, Z).

Conditional on the group, the probability density $f_{C_{j,g}}$ of the jth principal component of X is assumed to be the univariate Gaussian density with zero mean and variance $\lambda_{j,g}$. Under this model it follows that the unconditional approximated density of X can be written as

$$f_x^{(q)}(x;\theta) = \sum_{g=1}^{K} \pi_g \prod_{j=1}^{q_g} f_{C_{j,g}}(c_{j,g}(x);\lambda_{j,g})$$

where q_g is the number of the first principal components, $c_{j,g}(x)$ is the jth principal component score, with parameter $\theta = (\pi_g, \lambda_{1,g}, \ldots, \lambda_{q_g,g})_{1\leq g\leq K}$ and π_g are mixing probabilities ($\sum_{g=1}^{K} \pi_g = 1$).

Then the iterative Expectation-Maximization (EM) algorithm is used to maximize the pseudo completed log-likelihood:

$$L^{(q)}(\theta; X, Z) = \sum_{i=1}^{n} \sum_{g=1}^{K} Z_{i,g} \left(\log \pi_g + \sum_{j=1}^{q_g} \log f_{C_{j,g}}(C_{i,j,g}) \right)$$

where $C_{i,j,g} = C_{jg}(X_i)$ is the jth principal score of the curve X_i belonging to the group g.

Since the group label Z_i are unknown when estimate the mixture model parameters, the iterative EM algorithm[3] consists the computation for the con-

[3]The general Expectation-Maximization (EM) algorithm is widely applied to iteratively compute a maximum likelihood estimation for incomplete-data problems. In our study, it used for clustering when cluster labels are treated as "missing" values.

ditional expectation for the pseudo completed log-likelihood in the E step, and the computation of the principal component scores for each group in the M step.

3.3 Multidimensional Data Clustering in Frequency Domain

In time series analysis, any stationary process has both a time domain and a frequency domain representation. The dynamics of time series obtained from time domain can also be supplemented by frequency domain analysis, i.e. spectral analysis. On frequency domain, spectral analysis plays an important role in identifying and forecasting time series trend and analyzing the economic cycles.

In this section, we focus our attention on the frequency domain approach to clustering the commodity time series based on modeling spectral density functions. Firstly, spectral domain analysis of finite–dimensional stationary time series and the nonparametric technique for estimating spectral density is introduced. Then this method is extended to simultaneously estimate spectral density functions for a collection of stationary time series using a shared basis.

3.3.1 Spectral Analysis for Covariance-stationarity Process

Spectral analysis is concerned with exploration of cyclical patterns and is mainly used to detect the signal periodicities in the frequency domain by decomposing the process into an infinite sum of periodic functions, each having a different frequency ω ranging between 0 and π. For a covariance-stationarity process $\{X_t\}_{t=-\infty}^{\infty}$ the spectral representation can be written as

$$X_t = \mu + \int_0^{\pi} \alpha(\omega)\cos(\omega t)d\omega + \int_0^{\pi} \beta(\omega)\sin(\omega t)d\omega$$

where the frequency ω corresponds to a time horizon T, such as $T = 2\pi/\omega$, and weights $\alpha(\omega)$, $\beta(\omega)$ are random variables with zero mean.

Commonly used frequency estimation methods can be classified into parametric and nonparametric categories (see [32, 68, 112]). Parametric approach assumes that a certain model like ARMA model generates the signal and the pectral density (defined above) is computed from the ARMA model parame-

ters that are estimated by fitting the time series process. Nonparametric approach assumes the signal is composed of basis functions, such as discrete Fourier transform, and the signal at certain frequency is estimated by filtering or fitting. Functional structure estimation through nonparametric techniques has been developed in various areas, such as density estimation and regression.

Suppose that $\{X_1, X_2 \ldots, X_n\}$ is a zero–mean weakly stationary time series with autocovariance function $\gamma(h)$, where the hth autocovariance is defined as:

$$\gamma(h) = E(X_t X_{t+h}), \qquad h = 0, \pm 1, \pm 2, \cdots$$

If the sequence of autocovariances is absolutely summable[4], then it has the spectral representation as:

$$\gamma(h) = \int_{-1/2}^{1/2} f(\omega) e^{2\pi i \omega h} d\omega, \quad h = 0, \pm 1, \pm 2, \cdots$$

The autocovariance function and the spectral density function therefore form a Fourier pair, where $f(\omega)$ is the spectral density of X with the inverse Fourier transform:

$$f(\omega) = \sum_{h=-\infty}^{\infty} \gamma(h) e^{-2\pi i \omega h}, \quad -\frac{1}{2} \leq \omega \leq \frac{1}{2}$$

Then the periodogram can be denoted as $I_n(\omega)$ and it can be seen as a discrete Fourier transform of the sample covariance $\hat{\gamma}(h)$:

$$I_n(\omega) = \sum_{h=-(n-1)}^{n+1} \hat{\gamma}(h) e^{-2\pi i \omega h} = |\frac{1}{n} \sum_{t=1}^{n} X_t e^{-2\pi i t \omega}|^2$$

Now suppose X is a mean-zero stationary Gaussian process with a parametric covariance function $\gamma(h; \theta)$, then the likelihood function under Gaussian framework can be written as:

$$L(\theta) = \frac{1}{(2\pi)^{n/2} |\Gamma_{n,\theta}|^{1/2}} \exp\left(-\frac{1}{2} \mathbf{X}^\top \Gamma_{n,\theta}^{-1} \mathbf{X}\right)$$

where $\Gamma_{n,\theta}$ is the covariance matrix of the random vector X, then define

[4] A sequence $\{\gamma(h)\}$ is absolutely summable if it satisfies $\sum_{h=-\infty}^{\infty} |\gamma(h)| < \infty$.

$\ell(\theta) = -2 \times \log L(\theta)$:

$$\ell(\theta) \propto \log(|\Gamma_{n,\theta}|) + \mathrm{tr}(\mathbf{X}\mathbf{X}^\top \Gamma_{n,\theta}^{-1})$$

The Whittle log-likelihood function can be rewritten according to [121]:

$$\ell_W(\theta) = n \int_{-1/2}^{1/2} \left\{ \log(f_\theta(\omega)) + I_n(\omega) f_\theta(\omega)^{-1} \right\} d\omega \tag{3.5}$$

For a discrete frequency range, the Whittle approximation (3.5) has the following form:

$$\ell_W(\theta) = n \sum_{-1/2<\omega_j<1/2} \left\{ \log(f_\theta(\omega_j)) + I_n(\omega_j) f_\theta(\omega_j)^{-1} \right\} \tag{3.6}$$

3.3.2 Nonparametric Collective Spectral Density Estimation

Estimating spectral densities from high-dimensional time series is challenging. Current approaches are either not stable or computationally expensive. The nonparametric collective estimation that minimizes a penalized Whittle log-likelihood was developed by Maadooliat et al. [83] for multiple time series clustering, as the spectral density functions that share common features can be collectively estimated.

Given a collection of stationary time series $\mathbf{X}_i$'s, where $i = 1, \cdots, m$ and $X_i^\top = (X_{i1}, X_{i2}, \cdots, X_{in})$ with the associated spectral density f_i. assume each log-spectral density function can be represented by a linear combination of a common set of basis functions $\{\phi_k(\omega), k = 1, \cdots, K\}$ and its own set of coefficients. Specifically, we assume that $\log\{f_i(\omega)\} = u_i(\omega)$ with

$$u_i(\omega) = \sum_{k=1}^{K} \phi_k(\omega)\alpha_{ik}, \qquad i = 1, \cdots, m \tag{3.7}$$

The spectral density functions can be expressed as

$$f_i(\omega) = \exp u_i(\omega) = \exp\left\{ \sum_{k=1}^{K} \phi_k(\omega)\alpha_{ik} \right\}, \qquad i = 1, \cdots, m \tag{3.8}$$

The basis functions $\phi_k(\omega)$ are not pre-specified, thus need to be determined from the data. The dimension is reduced by representing the raw periodogram

on a set of common basis functions while allowing each one to differ by introducing random effects into the model.

To achieve this goal, the basis functions can be modeled by a low-dimensional subspace of a function space spanned by a rich family of fixed basis functions $\{b_\ell(\omega), \ell = 1, \cdots, L\}(L \gg K)$, such that

$$\phi_k(\omega) = \sum_{\ell=1}^{L} b_\ell(\omega)\theta_{\ell k}, \qquad k = 1, \cdots, K \tag{3.9}$$

For identifiability, $b_\ell, \ell = 1, \cdots, L$, are assumed linearly independent, i.e. the bases are orthonormal, and a large enough L ensures the needed flexibility in representing the unknown spectral densities. The most commonly used fixed basis functions are monomials, B-splines, Fourier, Polynomial and Wavelets. Bivariate splines can be used as the fixed basis functions for bivariate spectral densities.

Denote $\phi(\omega) = (\phi_1(\omega), \cdots, \phi_K(\omega))^\top$, $\alpha_{\mathbf{i}} = (\alpha_{i1}, \cdots, \alpha_{iK})^\top$, $\mathbf{b}(\omega) = (b_1(\omega), \cdots, b_L(\omega))^\top$, $\theta_k = (\theta_{1k}, \cdots, \theta_{Lk})^\top$, and $\mathbf{\Theta} = (\theta_1, \cdots, \theta_K)$. Then, the $u_i(\omega)$ in the vector-matrix can be simplified with eq(3.7) and eq(3.9) as

$$u_i(\omega) = \phi(\omega)^\top \alpha_{\mathbf{i}} = \mathbf{b}(\omega)^\top \mathbf{\Theta}\alpha_i, \qquad i = 1, \cdots, m.$$

The periodogram $I_{i,n}$ is a rough estimate of the spectral density, associated with the time series observations over n time points. Thus, for m spectral densities, the Whittle approximate has the form:

$$\ell_W(\mathbf{\Theta}, \mathbf{A}) = \sum_{i=1}^{m} \sum_{j} \{u_i(\omega_j) + I_{i,n}(\omega_j) \exp[-u_i(\omega_j)]\}$$

where

$$A = (\alpha_{\mathbf{1}}, ..., \alpha_{\mathbf{m}})^T.$$

Following [47] where a roughness penalty approach is used to estimate parameters by minimizing the penalized likelihood criterion $-2\ell_W(\Theta, A) + \lambda \sum_{k=1}^{K} \mathrm{PEN}(\phi_k)$, where $\mathrm{PEN}(\phi_k)$ is a roughness penalty function that ensures a smooth function by regulating the estimated basis function ϕ_k and $\lambda > 0$ is a penalty parameter. Newton-Raphson algorithm is used to minimize the penalized Whittle approximate log-likelihood.

3.4 Optimal Number of Clusters

The main challenge in clustering analysis is to find the optimal number of clusters, which is usually not predefined. Various techniques have been suggested to determine the optimal number of clusters [95, 44]. For example, BIC and AIC are frequently used in the context of traditional model-based clustering to select the number of clusters [1, 109, 16]. The ratio of the between and within-cluster sum of squares [21] and averaged Mahalanobis distance between the basis expansion coefficients and their closest cluster centre [111] are also calculated to choose the optimal number of clusters.

To select the optimal number of clusters in this study, we consider a popular technique known as the "elbow plot" or the "L-curve" [115], which uses the within-cluster dispersion to determine the number of clusters. To be more specific, L-curve looks at the percentage of variance explained as a function of the number of clusters and plots a sequence of the number of clusters C versus their corresponding within-cluster sum of squares. Within-cluster sum of squares is the summation of each clusters distance between that specific clusters with each points against the cluster centroid. Although this procedure is very simple and straightforward, it is not a formal statistical procedure. As the number of clusters increases, within-cluster sum of squares will decrease monotonically. However, the first value of the number of clusters at which within-cluster sum of squares reaches a minimum and remains stable indicates that there has been the largest increase in goodness of fit and hence which is the optimal number of clusters.

In our study, the number of clusters ranging from 1 to 12 are considered to permit exploring a wide range of possible number of clusters. Elbow plot in Figure 3.1 displays the optimal number of clusters is 3, based on the daily future data from the eighteen commodity products.

3.5 Clustering Results for Commodity Futures Prices

In this section, we will discuss the experimental outcome obtained from two model-based functional data clustering methods on both time domain and frequency domain. The data consists of the eighteen commodity products daily future price data according to Table 6.1, and the futures daily return are normal-

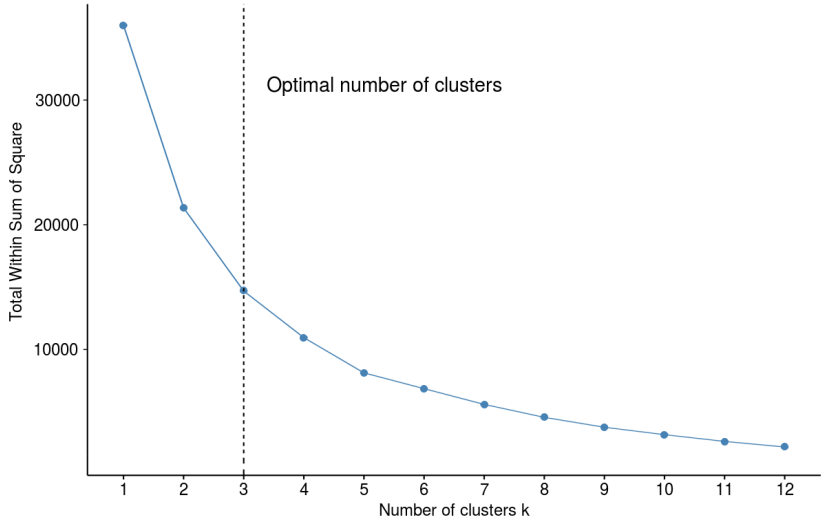

Figure 3.1: Elbow Plot for Optimal Number of Cluster

ized within the common time period from 01 January 2008 to 31 August 2018. The goal is to provide a classification results into 3 groups, and to compare between the two models.

For the time domain data clustering method, we first need to compute principal component functions for reconstruction of the functional form of the data. Figure 3.2 shows that the first four components account for 42%, 22%, 16% and 5% of the variation. For the clustering algorithm, we set up the maximum number of iterations as 200. Moreover, to avoid the convergence to a local maximum of the pseudo likelihood, the EM algorithm is designed to run 10 times with random parameter initializations. The whole fitting procedure is carried out through the *fda* package in R software.

For the frequency domain data clustering method, we can estimate smoothed spectral density functions in terms of frequency value. We choose B-spline as the fixed basis functions with a degree of 3 and the maximum number of iterations is fixed to 30. The algorithm was performed on a web application available at "https://ncsde.shinyapps.io/NCSDE".

We attempt to find the optimal combination between the classifier and the number of features that are fed to it. The number of iterations for both methods is fixed to 500, meaning we design an experiment whereby we attempt each method for 500 times on identical datasets. The Figure 3.3 illustrates the sim-

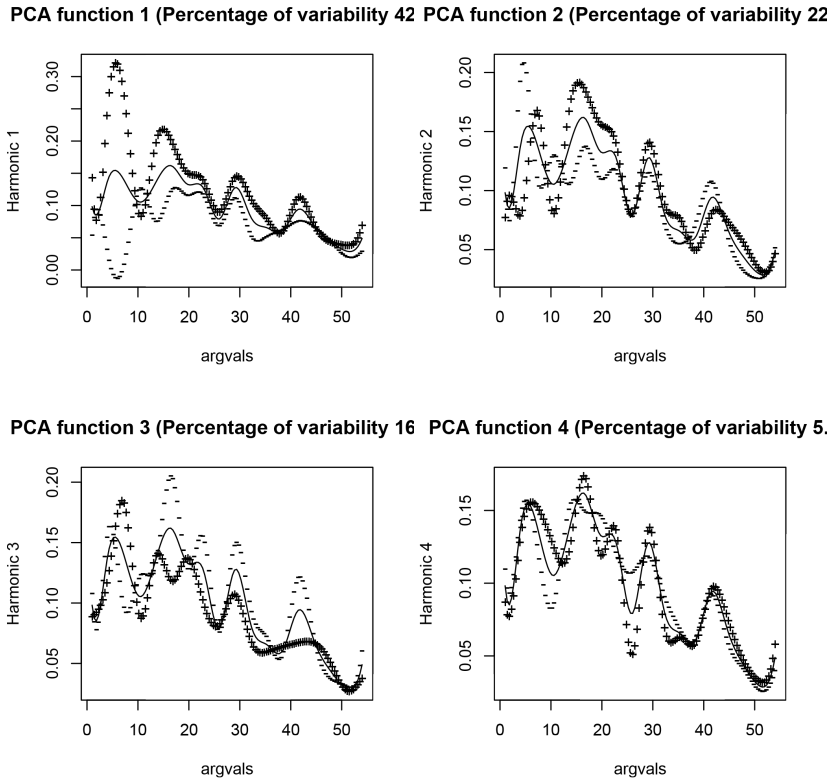

Figure 3.2: First Four PCA Function Plot

	Time Domain Cluster	**Frequency Domain Cluster**
Group 1	wheat, oats, canola	oats, aluminum, copper, palladium
Group 2	soybean oil, soybean, corn, sugar, orange juice, cocoa	orange juice, coffee, sugar, crude oil
Group 3	coffee, aluminum, gold, copper, palladium, platinum, lean hogs, feeder cattle, crude oil	wheat, soybean oil, soybean, corn, canola, cocoa, gold, platinum, lean hogs, feeder cattle

Table 3.1: Clustering Results between time domian analysis and frequency domian analysis on commodity Future price

ilarities and the differences between the different curves of commodity data. Table 3.1 shows the top clustering result from two methods, both obtaining an accuracy of 74% and over [5].

[5]One should note that other cluster results were evaluated in the same test, but failed to

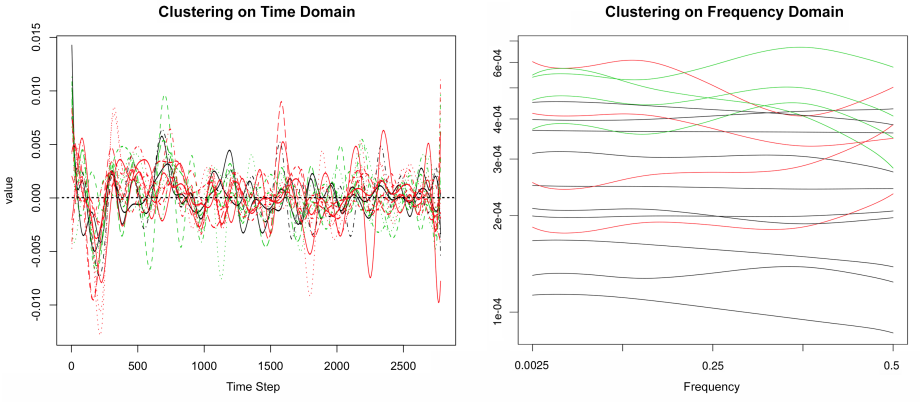

Figure 3.3: Commodity Future Price Clustering Results based on Time Domain Analysis (left) and Frequency Domain Analysis (right).

The results show that two clustering methods provide different compositions of clusters. For the time domain data clustering method, it is hard to observe three distinct groups from the left graph in Figure 3.3. We can see Group 1 only contains the grains products while Group 2 contains the rest of grains and all soft products, expect Coffee - which are part of the Group 3 with all the metals, livestock and energy products. However, from the right graph the curves in the same cluster have similar shape. We can observe three distinct groups of commodity products, although each group contains mixed of products from five main categories. The green group tends to decrease with high frequency while the red group performs oppositely and the last group (black) mostly stay the same. Normally, clustering method should yield similar results from the same stationary time series data regardless its time domain presentation or frequency domain presentation. This is due the the characteristics of the data are fundamentally the same for both presentation method. However, we do observe different grouping results from our research and this will be further studied in future researches. We cannot provide a definite explanation here but suspect the grouping difference is due to long-memory processes that are characterized by auto-covariance sequences that are square summable but not absolutely summable. Such models may have spectra that are unbounded at the zero frequency.

There are many standard methods to evaluate the performance of the cluster

provide more stable results, hence are omitted from this table.

results in discovering the correct group label (see [5, 8]) but these strategies are criticized as misleading since they rely on the assumption that class labels occur simultaneously with cluster structure. Kogan [72] discussed that the best way to evaluate clustering performance is to explain how the clusters make sense on real-world datasets. Hence, the evaluation strategy in this study is to model the impact of clustering results on forecast abilities. In other words, the cluster that gives us the most appropriate input information that can provide the most accurate forecast results will be the best model. This strategy will be analyzed in the next chapter.

Chapter 4

Long Memory Time Series Forecasting

4.1 Development and Application of Machine Learning

Artificial intelligence (AI) is the study of making machines or software capable of intelligent behaviors. The ideas behind it inspire many scientists to begin critically discussing the possibility of building an electronic brain. The beginning of modern AI was originated in classical philosophers' attempts to describe human thinking as a mechanical manipulation of symbols since antiquity. Early AI researchers developed algorithms that help people to solve puzzles or make logical deductions. Later on, AI has gain the ability to identify and distinguish between different objects, categories, relations and situations. Many researchers like Laird et al. [73] and Geary [41] believe that their work needs to be incorporated into a machine with general intelligence (known as strong AI), combining all the skills and even exceeding human abilities at most or all of them. Machine learning, the study of computer algorithms, is another key research field in AI that let the computer have the ability to learn and improve automatically through experience. Mathematical analysis of machine learning algorithms are well developed by scientists and their performance are continuously improved, for example, Natural language processing gives machines the ability to read and understand the languages that humans speak [92].

Today, Artificial Intelligence has been a thriving field with many practical applications and active research topics applied that can solve real-life problems, like spam emails classification, face recognition, image classification, speech recognition, signal denoising, weather forecast, robot design and so on, see[19, 20, 107] for more. Over the last few years, artificial neural network and deep learning have gained extensive attention. The study of artificial neural networks began a decade before, Rosenblatt [105] invented the perceptron that used input from sensors such as cameras, microphones to deduce aspects of the world. Werbos [119] further developed the backpropagation algorithm. The main categories of networks are feedforward neural networks and recurrent neural networks. Furthermore, deep neural network is built as an artificial neural network with multiple hidden layers between the input and output layers. As a result, this kind of Artificial Intelligence can be applied to not only the learning part, but also the problem of intelligent control. But on the other hand, advances machine learning algorithms and computer hardware are needed for more efficient methods for model training and control, see [113] for detials.

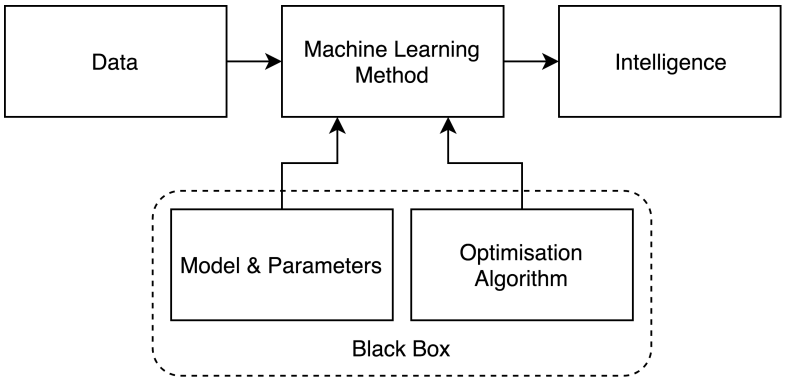

Figure 4.1: We developed this graph to illustrate the Machine Learning process

In recent years, "Machine Learning" has become more and more popular, as it concerns the design and development of algorithms that allow computers to evolve behaviors based on empirical data and provides computers with the ability to learn, see Figure 4.1. By gathering knowledge from experience, machine learning allows the computer to learn complicated concepts by building algorithms out of simpler ones without explicit operations form human beings.

There are several different categories of Machine Learning, for instance, supervised learning, unsupervised learning and reinforcement learning. Supervised learning includes both classification and regression, where classification is to arrange a group of things in classes or categories according to different characteristics based on a number of examples of several categories. Regression is the fitting of a function that explains the relationship between inputs and outputs for the prediction. Unsupervised learning is finding patterns in a stream of inputs, such as clustering or image identification. Reinforcement learning is to make sequence decisions that enable an agent to automatically determine the behavior in an interactive environment by trial and error using feedback from actions and experiences, in order to maximize its performance.

4.2 Artificial Neural Networks and Learning Algorithms

Although many economic data can be adequately explained by linear mathematical models, they usually can be better modeled using nonlinear model in nature, especially when used for future prediction. On the other hand, the increased volatility in commodity prices also makes the forecasting more difficult since the traditional methods are less reliable and complex forecast methods are not robust in today's market environment. To overcome these limitations, machine learning algorithms such as the artificial neural networks (ANNs), support vector machines (SVMs) and relevance vector machines (RVMs) are proved to be useful as an alternative to forecast financial data and are able to capture the market pattern (see [120, 37, 22, 69, 59]).

Artificial neural networks are a class of generalized nonlinear nonparametric models based on studies of the human brain activities and are able to learn. From a learning process, they get the intelligence which can mine valuable information from a mass of historical data. In recent years, machine learning technique, especially artificial neural networks are widely used in financial data mining or forecasting. For example, Zhang et al. [124] showed that a modified neural network forecasting model was able to perform with high accuracy, and the mining system had a better capability of controlling risk. In [2] a Neural Network model was developed for forecasting stock price time series data regarding Intercontinental Bank Nigeria.

4.2.1 Multilayer Perceptron (MLP)

In machine learning, multilayer perceptron (or MLP) is the classical type of feedforward artificial neural network and are used frequently in time series prediction. The structure of layered feedforward neural networks is shown in Figure 4.2.

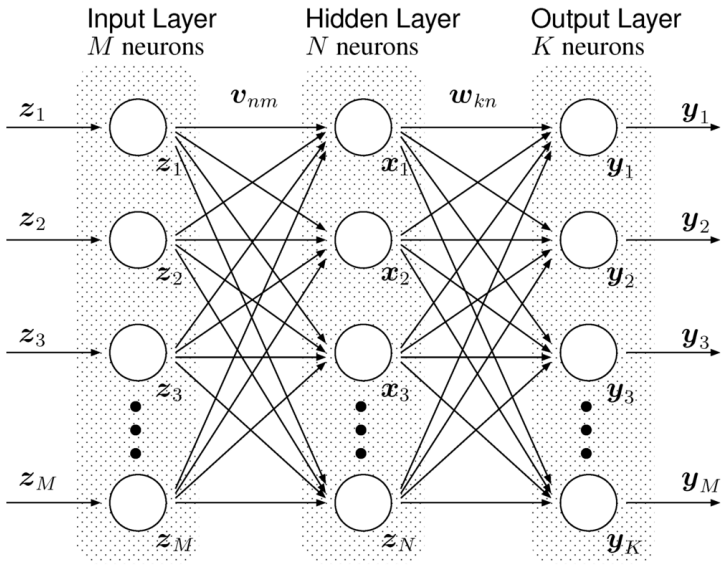

Figure 4.2: Perceptron Neural Networks

Each of these networks consists of a set of inputs and one or more layers of parallel neurons. These networks are called neural because they are loosely inspired by neuroscience. Inputs are connected only to neurons in the first layer (referred to as an input layer) usually with an extra input b_0 representing the bias of each neuron. Neurons in one layer are connected to all neurons in the hidden layer. The last layer, called output layer, producing the output of the network, is called an output layer. Any layers that precede the output layer are called hidden layers. Each hidden layer of the network is typically vector-valued. The dimensionality of these hidden layers determines the width of the model. Basic Neural Network consist of three layers: input, hidden and output layer, and the neurons in each layer are interconnected by connection strengths called weights.

4.2.2 Backpropagation (BP) Algorithm

MLP utilizes a supervised learning technique called backpropagation for training. This is a supervised learning technique which is based on the Gradient Descent method that attempts to minimise the error of the network by moving down the gradient of the error curve. Idea behind BP algorithm is quite simple; the training process can de described as: for each input entry in the training data set, input signals are forward-propagated through the network towards the output, initialized weights and output against desired value are checked and feed back with errors. When output of Neutral Network is evaluated against desired output, the weights between layers are modified by using a gradient descent training algorithm and updated until error is small enough.

The backpropagation algorithm was developed by Werbos [119] and rediscovered independently by Priddy and Keller [101]. It is widely used as the most popular, effective, and easy-to-learn model in feed forward multilayer neural networks and it performs parallel training for improving the efficiency of Multilayer Perceptron network. It is considered a generalization of the delta rule for nonlinear activation functions and multi-layer networks [119].

If we associate index m with the input layer, index n with the hidden layer, and index k with the output layer, then an output unit in the network diagrammed in Figure 4.2 computes an output value y_k given an input $z_m$via the following compositional function:

$$y_k = g_k(b_k + \sum_n g_n(b_n + \sum_m z_m w_{mn})w_{nk})$$

where g_k is the activation function for units in that layer k, and the weight $w_{m,i}$ links the outputs of units feeding into layer i to the activation function of its units . The term b_i is the bias for units in layer i. One training method is to determine the network weights that minimize the errors the network makes. A standard way of quantifying error is to take the squared difference between the network output y_k and the target value t_k:

$$\theta = \frac{1}{2} \sum_{k \in K} (y_k - t_k)^2$$

One method to find optimal weights is gradient descent method. δ is defined as the first-order derivative of total error function such that $\delta = \partial\theta/\partial w$. The update rule of the algorithm could be

$$w_j := w_j - \alpha\delta_j$$

where α is the learning rate.

Another widely used method is Newton's method, where the second-order derivatives of the total error function are calculated for each component of gradient vector. In order to get the minima of total error function θ, each element of the gradient vector should be zero. To be more specific, assume N is the total number of output nodes, and the gradient components we have can be denoted by $\delta_1, \delta_2, \ldots, \delta_N$ that can be written in matrix form:

$$\begin{bmatrix} -\delta_1 \\ -\delta_2 \\ \vdots \\ -\delta_N \end{bmatrix} = \begin{bmatrix} \frac{\partial^2\theta}{\partial w_1^2} & \frac{\partial^2\theta}{\partial w_1 \partial w_2} & \cdots & \frac{\partial^2\theta}{\partial w_1 \partial w_N} \\ \frac{\partial^2\theta}{\partial w_2 \partial w_1} & \frac{\partial^2\theta}{\partial w_2^2} & \cdots & \frac{\partial^2\theta}{\partial w_2 \partial w_N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2\theta}{\partial w_N \partial w_1} & \frac{\partial^2\theta}{\partial w_N \partial w_2} & \cdots & \frac{\partial^2\theta}{\partial w_N^2} \end{bmatrix} \times \begin{bmatrix} \Delta w_1 \\ \Delta w_2 \\ \vdots \\ \Delta w_N \end{bmatrix}$$

where the square matrix is called Hessian matrix such that

$$H = \begin{bmatrix} \frac{\partial^2\theta}{\partial w_1^2} & \frac{\partial^2\theta}{\partial w_1 \partial w_2} & \cdots & \frac{\partial^2\theta}{\partial w_1 \partial w_N} \\ \frac{\partial^2\theta}{\partial w_2 \partial w_1} & \frac{\partial^2\theta}{\partial w_2^2} & \cdots & \frac{\partial^2\theta}{\partial w_2 \partial w_N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2\theta}{\partial w_N \partial w_1} & \frac{\partial^2\theta}{\partial w_N \partial w_2} & \cdots & \frac{\partial^2\theta}{\partial w_N^2} \end{bmatrix}$$

and then more formally, we can obtain

$$-\delta = H\Delta w$$

Therefore, the update rule for Newton's method is

$$w_j := w_j - H_j^{-1}\delta_j$$

4.2.3 Levenberg-Marquardt (LM) Algorithm

The Levenberg–Marquardt algorithm blends the gradient descent method and the Gauss-Newton algorithm. It's more robust than the Gauss-Newton algorithm, because it can converge well even if the error surface is much more complex than the quadratic situation [97]. The basic idea of the Levenberg–Marquardt

algorithm is that it performs a combination of the gradient descent algorithm and the Gauss–Newton algorithm: during the training process, the Levenberg–Marquardt algorithm switches to the gradient descent algorithm, until the local curvature is appropriate to make a quadratic approximation; then it approximately becomes the Gauss–Newton algorithm, which can speed up the convergence significantly.

In order to make sure that the approximated Hessian matrix is invertible, Levenberg–Marquardt algorithm introduces another approximation to Hessian matrix:

$$H \approx J^T J + \mu I$$

where J denotes the Jacobian matrix $J_{ij} = \partial e_i / \partial w_j$ in Gauss–Newton algorithm, e is a vector of total error function θ, μ is the combination coefficient that is always positive and I is the identity matrix. With this approximation,matrix H is always invertible since the elements on the main diagonal of the approximated Hessian matrix are larger than zero with the positive μ. Thus, the update rule of Levenberg–Marquardt algorithm can be presented as

$$w_j := w_j - (J^T J + \mu I)^{-1} J_j e_j$$

In fact, when the combination coefficient μ is zero, this is just Newton's method (also called Newton–Raphson method) using the approximate Hessian matrix; when μ is very small (nearly zero), it is approaching to Gauss–Newton algorithm; but when μ is large, this becomes gradient descent with a small step size, which can be interpreted as the learning coefficient $\alpha = 1/\mu$. Although the Levenberg-Marquardt algorithm tends to converge a bit slower than Gauss–Newton algorithm, it converges much faster than the gradient descent method.

4.3 Recurrent Neural Networks (RNN)

Recurrent neural network (RNN) is a type of artificial neural network designed to recognize patterns in sequences of data, such as text, images, sensors and stock markets. Typically a RNN approach is based on learning from sequences and the networks in loop allow the information to persist. Each network in the loop takes input and information from previous network, so that sequential

information is preserved in the recurrent network's hidden state, which manages to span many time steps as it cascades forward to affect the processing of each new example.

The purpose of this chapter is to design a recurrent neural network that is based on a nonlinear autoregressive network with exogenous inputs (NARX) model, while taking into account the clustering features.

4.3.1 Nonlinear Autoregressive Network with Exogenous Inputs (NARX)

The recurrent dynamic network used in our research is a nonlinear autoregressive network with exogenous inputs (NARX) model, which allows the output time series to be related to its previous data and another independent time series, i.e. the data in the same cluster. Based on different cluster results in the previous subsection, we use different input time series corresponding to each method.

The NARX model is based on the linear ARX model, which is commonly used in time series modeling. Compared with common multi-layer perceptron network, the model we used is a typical recurrent neural network or RNN introduced by Rumelhart et al. [106].

From multi-layer networks to recurrent networks, a related idea we need to mention is the use of convolutional approach, that is, the basis for time-delay neural networks [74]. The output of convolution is a sequence where each member of the output is a function of a small number of neighboring inputs, supplying neural networks with "memory" to deal with the temporal dimension. This NARX feedback neural networks with feedback connections enclosing several layers allow the output signal of a time series y_t to relate to previous data of itself and another independent time series x_t.

Definition 9. The defining equation for the NARX model is

$$y_t = f(y_{t-1}, y_{t-2}, \cdots, y_{t-n_y}, x_{t-1}, x_{t-2}, \cdots, x_{t-n_x}) \tag{4.1}$$

where the next value of the dependent output signal y_t - the predicted value of y for the time t, is regressed on previous values of the past output signal $y_{t-1}, y_{t-2}, \cdots, y_{t-n_y}$ and previous values of exogenous inputs $x_{t-1}, x_{t-2}, \cdots, x_{t-n_x}$. To implement NARX model we can approximate the mapping function $f(\cdot)$ of the neural network following a feedforward neural network. Here n_x is the number of input delays and n_y is the number of output delays.

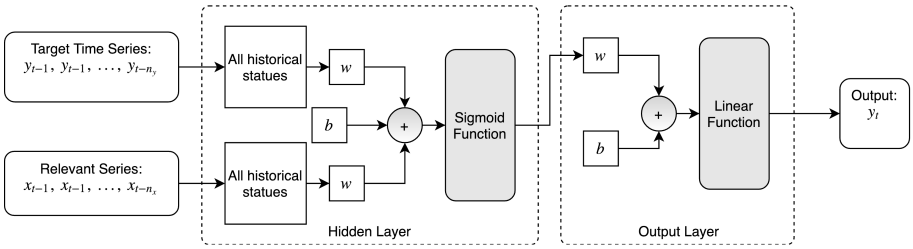

Figure 4.3: Nonlinear Autoregressive Network with Exogenous Inputs (NARX) Architecture

Figure 4.3 illustrates the model architecture in NARX design. The network is first trained by the input series $x_{t-1}, x_{t-2}, \cdots, x_{t-n_x}$ and target time series $y_{t-1}, y_{t-2}, \cdots, y_{t-n_y}$. In our research study, the price of a specific product can be affected by other products directly or indirectly. Assuming that our model consists of a deterministic component and a statistical component, then we can have two inputs: 1) for the deterministic component, the daily price data of a single product we target for forecast will simply be included as endogenous input of the NARX model. 2) for the statistical component, the exogenous input contains the daily price data of other products that are relevant to the target, i.e. the products in the same cluster as the predicted one. More specifically, we denote the model as NARX_{Time} and $\text{NARX}_{Frequency}$ when the exogenous inputs chosen from the cluster results based on either time domain clustering or frequency domain clustering, respectively.

During the training phase the true past values are fed into the network to increase the model accuracy. The time series in the same cluster will be the second type of entry as exogenous input signals in the model. In general, if we have K commodity variables in the same cluster group g, then the exogenous inputs is a matrix denoted as

$$\begin{aligned}\mathbf{X}^g &= \begin{bmatrix} \mathbf{x}^g_{x-n_x} & \mathbf{x}^g_{x-n_x+1} & \cdots & \mathbf{x}^g_{t-2} & \mathbf{x}^g_{t-1} \end{bmatrix} \\ &= \begin{bmatrix} x^g_{1,x-n_x} & x^g_{1,x-n_x+1} & \cdots & x^g_{1,t-2} & x^g_{1,t-1} \\ x^g_{2,x-n_x} & x^g_{2,x-n_x+1} & \cdots & x^g_{2,t-2} & x^g_{2,t-1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x^g_{K,x-n_x} & x^g_{K,x-n_x+1} & \cdots & x^g_{K,t-2} & x^g_{K,t-1} \end{bmatrix}\end{aligned}$$

The mapping function $f(\cdot)$ is initially unknown and it is approximated during the training process. The internal architecture that performs this approximation is purely feedforward network, meaning the usual training algorithm the Levenberg-Marquardt (LM) algorithm, can be used. Similar to the MLP, the weights can be adjusted so that the NARX model produce an output close to the target values.

After the training process, the model with the lowest out-of-sample testing error is chosen for forecasting, a network is simulated in open-loop form, which replaces its associated feedback layer weights with a new input and input weight connections, for as long as output data is known. To perform multi-step prediction it switches to closed-loop form which replaces its associated feedback input and their input weights with layer weight connections coming from the output.

4.3.2 Long Short-Term Memory (LSTM)

The mathematical challenge of learning long-term dependencies in recurrent networks is the vanishing gradient problem. As the length of input sequence increases, the gradient signal gets so small that learning becomes very slow for long-term dependencies, and it becomes harder to capture the influence of the earliest time.

Long Short-Term Memory (LSTM) was developed by Hochreiter and Schmidhuber [56]. With a special RNN structure, it is proven to be stable and powerful when predicting time series with long-range dependencies [43, 42, 84].

Comparing to RNNs, LSTM's cell has a more complex structure. The hidden layer(s) contains the memory cell, which is the unique part of LSTM, containing three gating units, i.e. the input gate i_t, the forget gate f_t and the output gate o_t. Both input and output gates are the same as the RNN's input and outputs layers with corresponding weights. The forget gate learns how to remember or forget its previous state, allowing LSTM to catch more complex temporal patterns. Through the three gates in each of the memory cells, the network maintains and adjusts its cell state:

$$
\begin{aligned}
i_t &= \sigma(W_i x_t + U_i h_{t-1} + b_i) \\
f_t &= \sigma(W_f x_t + U_f h_{t-1} + b_f) \\
o_t &= \sigma(W_o x_t + U_o h_{t-1} + b_o) \\
c_t &= f_t c_{t-1} + i_t \cdot tan\, h(W_c x_t + U_c h_{t-1} + b_c) \\
h_t &= o_t \cdot tan\, h(c_t)
\end{aligned}
$$

where x_t is the input to the memory cell, σ is the sigmoid active function, W_i, W_f, W_c, W_o, U_i, U_f, U_c and U_o are the weight matrices, b_i, b_f, b_c and b_o are biases, i_t, f_t, o_t represent the values of input, forget and output gate, c_t maintains state from time step to time step and h_t represents the output of hidden state.

4.4 Network Training and Implementation

In order to implement prediction results, we are going to train four machine learning models: multilayer perceptron (MLP) model, nonlinear autoregressive network with exogenous inputs from time domain clustering (NARX_{Time}) model, nonlinear autoregressive network with exogenous inputs from frequency domain clustering ($\text{NARX}_{Frequency}$) model, and long short-term memory (LSTM) model. This section includes detailed descriptions of the following five steps and implementation results.

Step1: Data Processing

All the commodity future price data are collected from the Bloomberg database and pre-processed for the networks. Detailed data description is demonstrated in the first chapter, and all the data are collected within the same time frame from 01 January 2008 to 31 August 2018 for consistency. Since some commodity input with limited available data points will have impact on the training set and neural network performance, all missing values are interpolated linearly to keep as many training examples as possible.

Although it is commonly accepted that the more input variables fed in the network the better results one can get, research shows that the predictive power will decrease if too much information is fed. This is due to the additional noise being increased when more data sets are fed to the model [37]. Hence, in order

to reduce the number of input variables that are irrelevant but keep as much information as possible, the inputs used in the two NARX models are applied the three optimized clustering groups shown in the last chapter, since commodity time series in the same cluster represent the high internal relationship that can improve the predictive power.

Step 2: Training, Validation and Testing Sets

The data set is divided into three subsets: 1) Training sets: for training the network to recognize patterns in the data. 2) Validation sets: for the model selection in order to choose the configuration that has the best predictive power on the specific time series. 3) Testing sets: for model evaluation. In our study, we randomly divide all the historical data into three groups: 70% for the training set, 15% for validation set and the remaining 15% for the testing set.

Step 3: Neural Network Paradigms

The network is divided into three layers: input layer, hidden layers and output layer. Although increasing the number of hidden layers provides the ability to generalize, in practice, a network with two hidden layers and sufficient number of nodes is enough and has historically shown good performance [65]. In our model, we have chose to use one output layer and one input layer and L is defined to be from 2 to 5 as the reasonable range of hidden layers. A hidden layer optimization test is automatically implemented to evaluate the optimal number of hidden layers for each model with the top prediction performance.

As well known, the performance of neural networks depends on a number of subjective choices, for example, the choice of the architecture (layers, units, transfer functions), the choice of the training algorithm, and the choice of the initial weights. To keep the performance comparable where no method obtains the computation advantage of the other, besides the same separation ratio of training, validation and testing, we used the same or similar parameters in the hidden parameters for all the forecasting models.

Step 4: Neural Network Training

In order for a network to be able to recognize patterns it needs to be presented with observations from the training data set as paired inputs and outputs, called supervised learning. This allows computation of the optimal weights between neurons. We first initialize all the weight matrices with random values, and

carry out training with an optimization heuristic to find the set of weights that minimizes the error function. We apply backpropagation Levenberg–Marquardt learning algorithm to train our neural network model as it converges much faster than the common stochastic gradient descent method. As the learning rate is adaptive to the error function, the initial choice is set to the standard value 0.01. If the error function decreases the learning rate will be increased, and vice versa.

Once the networks are trained, they are tested on validation data sets. This allows comparison of estimated out-of-sample predictive performance of different architectures. The network that has the best predictive performance on the training sets for every time series is assumed to have the best generalization ability. Then this network is used for evaluating the neural network technique on the test set.

Step 5: Implementation

In our study, lags of 30 time steps are chosen to present a month of data taken from the previous time steps, to use as input variables predicting the next time period. For model hyper parameters, the same batch size[1] of 16 is used by the stochastic gradient descent in each step. Due to the fact that real commodity price data is available on the market on daily basis, one can have access to the actual values of time steps between prediction. Our neural network state automatically renews everyday the latest available input data with the observed values instead of the predicted values.

One key thing that remains to be determined is the number of epoch - one complete presentation of the data set to be learned to a learning machine. To ensure we have a convergence for the error term and over-fittings are adequately avoided, a loop of multiple choices of epochs is taken from 2^i with $i = 1, 2, 3, ..., 12$. By comparing the error term for out-of-sample testing, we then selected the optimal epoch number to train and use in the model for each corresponding data set. Based on the optimal epoch number chosen, we run the model based on training data and validation data to predict the testing data range one step at a time with the real data being added dynamically as well. For the actual implementation of both the multilayer perceptron and the recurrent neural network, Tensorflow is used via the Keras framework in Python, while for NARX neural network we implement through Matlab Neural Network Toolbox.

[1]The batch size is a number of samples processed before the model is updated

For comparison purposes, ARIMA model is also added in this test to provide benchmarks for all error measures. There are two practices an ARIMA model can employ in this case, one way is to update the ARIMA model at each time step with the newly available data when forecast the next step. In this sense, the ARIMA model is alway re-trained and provides a one step forecast essentially. Another practice is using the training data sets to estimate the ARIMA model parameters and make the forecast based on these estimated parameters. While the first method may provide more accurate forecast results due to the frequent updates, the second method is closer to the machine learning model structures. In this research, since the ARIMA models are not used for goodness of fit comparison, we choose the first method to provide better error measurement statistics for benchmarking purpose only. The best ARIMA model for each individual commodity is selected by looping over a large set of ARIMA models and the model that generates the best information criterions will be used.

More specifically, as show in Table 6.9 in Appendix, a set of ARIMA models are pre-defined. These are ARIMA models with AR lags as 2, 4 or 8, with MA lags as 0, 1 or 2 and finally with integral equals to 0 or 1 indicating whether the first differences is taken into the data. The system has a total of 18 ARIMA models with all different AR, MA combinations. For each product, all 18 models are estimated and their information criterions[2] are calculated. Finally, the best ARIMA model configuration is selected by adopting the ARIMA model with the lowest information criteria values.

Model Forecasting Results

The plots Figure 4.4 and 4.5 provide multistep prediction results of daily future price for all the eighteen commodities[3] starting from 01 January to 31 August in year 2018. This 8 months observation window provides us a decent size for model forecast-ability testings. In each figure, the yellow line represents forecast results using multilayer perceptron (MLP) model, purple line and green line denote results from nonlinear autoregressive network with exogenous inputs from time domain clustering (NARX_{Time}) model and frequency domain clustering ($\text{NARX}_{Frequency}$) model, and blue line stands for long short-term memory

[2]We use the summation of AIC (Akaike information criterion), BIC (Bayesian information criterion) and HQIC (Hannan–Quinn information criterion) as the finally information criteria value. See Box-Muller method in [17] for details.

[3]These commodities are wheat, soybean oil, soybean, oats, corn, canola, sugar, orange juice, cocoa, coffee, aluminum, gold, copper, palladium, platinum, lean hogs, feeder cattle and crude oil.

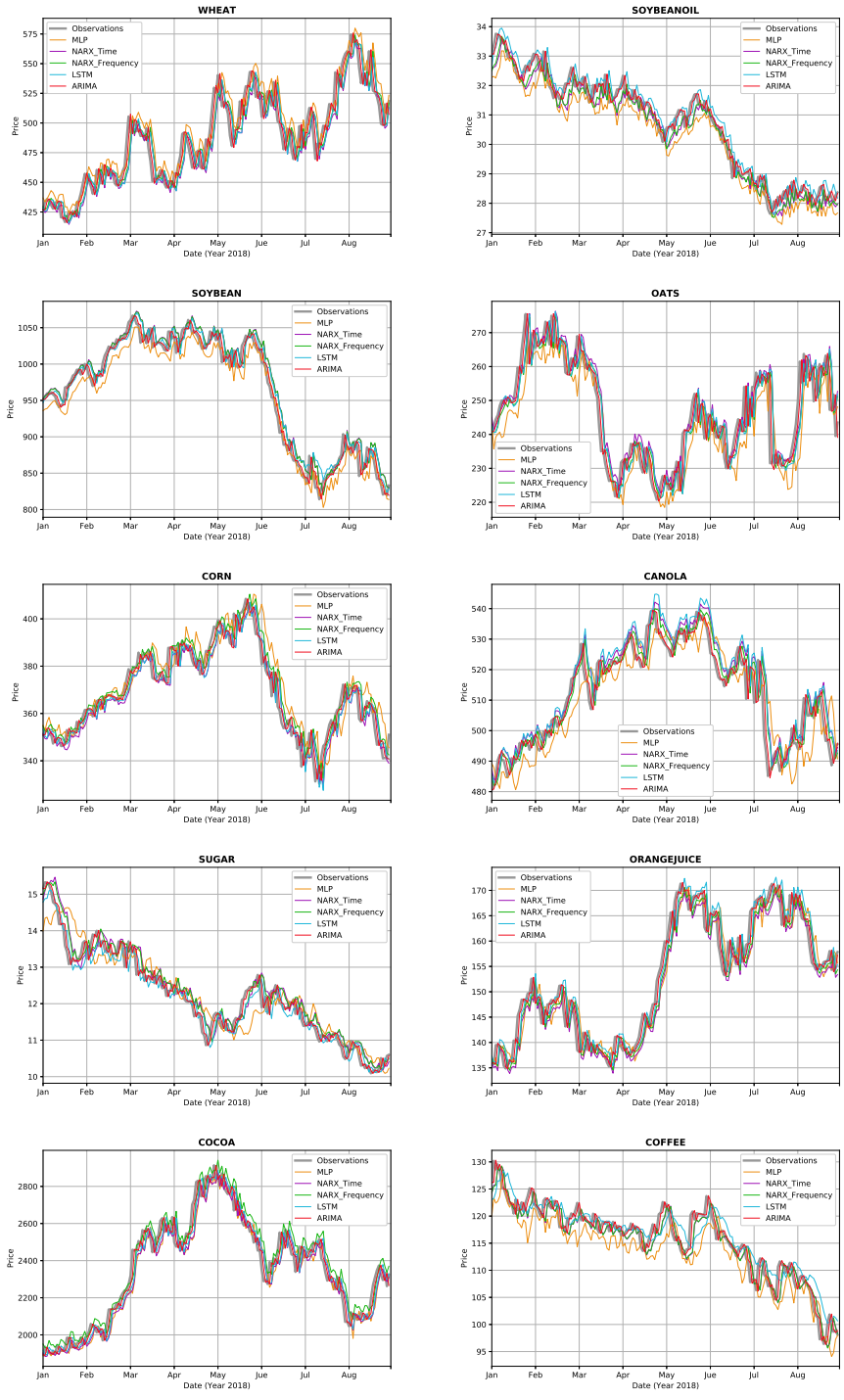

Figure 4.4: Community Future Forecast Result. Comparison based on ARIMA, MLP, NARX$_{Time}$, NARX$_{Frequency}$ and LSTM Model.

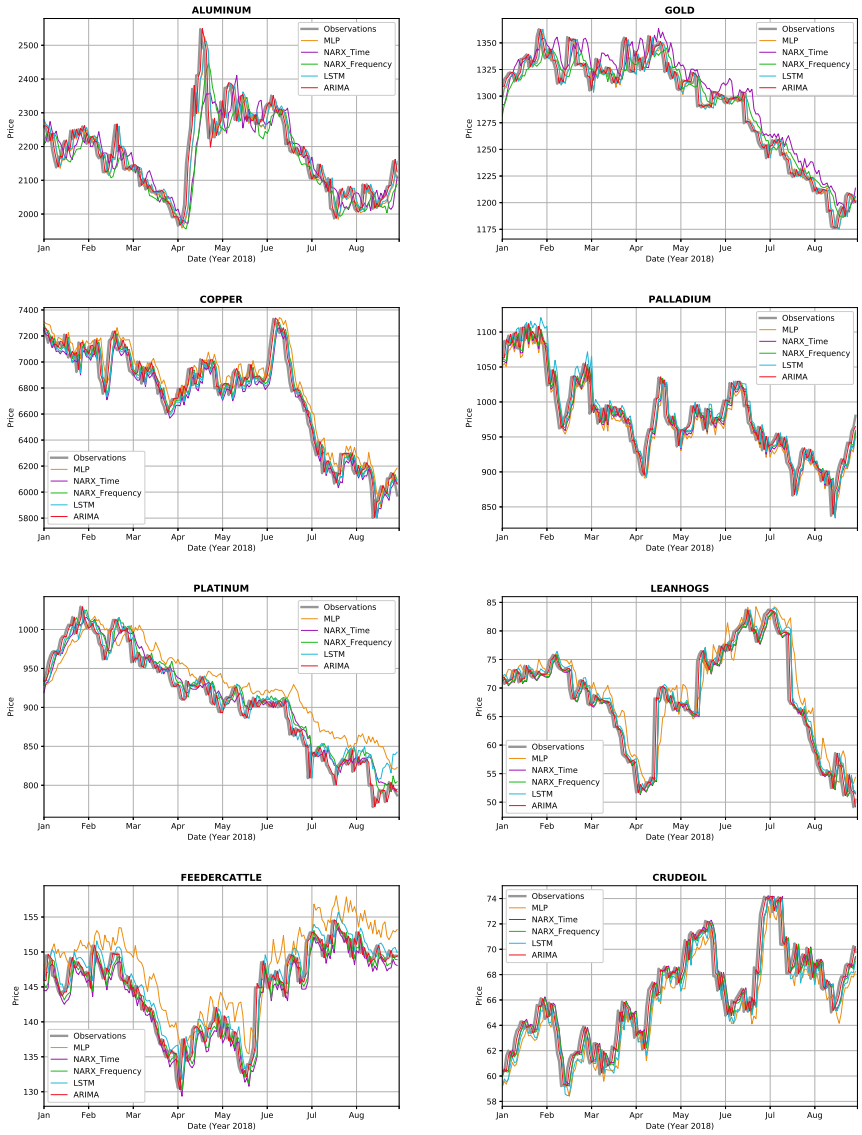

Figure 4.5: Community Future Forecast Result. Comparison based on ARIMA, MLP, NARX$_{Time}$, NARX$_{Frequency}$ and LSTM Model.

(LSTM) model prediction results. In addition, to compare the machine learning forecasting results with other models, the autoregressive integrated moving average (ARIMA) model (plotted as red line) is used as the benchmark in this research. The true observation are also plotted in gray line with four machine learning forecast results as comparison .

The performance results on all products clearly shows that all the machine learning algorithms are able to capture nonlinear relationship as well as performing the forecast, especially the ability of identifying trend patterns. In contrast, the ARIMA model needs to be updated at each step in order to provide some adequate forecast. This concludes that through the learning process, neural networks with more clustering information are capable of capturing nonlinear relationship as well as performing the forecast, especially the ability of identifying trend patterns.

4.5 Measure the Model Performance

In theory, the model's performance in the validation period is the best guide to its ability to predict the future. The evaluation indicates the true out-of-sample performance of autoregressive integrated moving average (ARIMA) models, nonlinear autoregressive network with exogenous inputs from time domain clustering ($\text{NARX}_{Frequency}$) and frequency domain clustering ($\text{NARX}_{Frequency}$). As we are interested to evaluate whether our model is suitable for estimating the expected return, multilayer perceptron (MLP) model and long short-term memory (LSTM) model are used for model comparison.

4.5.1 R Squared (R^2)

In order to measure the goodness of our fit, a few measurements are proposed here. R squared is a common statistical measure of how close the data are to the fitted with the regression results. It can be calculated by dividing explained variation with total variation. More specifically, given a set P comprising pairs of the true values (or targets, denote y_k) and predicted (fitted) values ($\hat{y}_k$), one can work out the R^2 by

$$R^2 = 1 - \frac{\sum_{k \in P}(y_k - \hat{y}_k)^2}{\sum_{k \in P}(y_k - \bar{y}_k)^2}$$

where $\bar{y}_k = \frac{1}{N}\sum_{k\in P} y_k$ is the data mean, and N is the set size of P.

4.5.2 Normalized Mean Squared Error

One usual measure to evaluate and compare the predictive power is the normalized mean squared error (NMSE). Given a set P comprising pairs of the true values (or targets, denote y_k) and predicted values ($\hat{y}_k$), the NMSE which normalizes the MSE by dividing it through the variance of respective series can be defined as

$$NMSE = \frac{\sum_{k\in P}(y_k - \hat{y}_k)^2}{\sum_{k\in P}(y_k - \bar{y}_k)^2} = \frac{1}{\sigma_P^2}\frac{1}{N}\sum_{k\in P}(y_k - \hat{y}_k)^2$$

where σ_P^2 is the estimated variance of the data, $\bar{y}_k$ being the mean, and N is the size of a set P. Note, one can have $NMSE = 1 - R^2$.

4.5.3 Mean Absolute Percentage Error

Another common measure of forecast error is called mean absolute percent error (MAPE, [70]), which is popular due to its advantages of scale-independency and interpretability. Following the same notation, MAPE can be obtained by

$$MAPE = \frac{1}{N}\sum_{k\in P}\frac{|\hat{y}_k - y_k|}{y_k}$$

the absolutely difference between predicted values ($\hat{y}_k$) and actuals (y_k) is calculated and then divided by actuals to gives the percentage per observation error. An average is then applied on all the observations errors to provide an overview of absolute changes.

4.5.4 Directional Change

Since the normalized mean square errors (NMSE) and mean absolute percentage error (MAPE) measure only provide measures on the scale or size of the errors, we employed another error measure to test the direction forecast-abilities. The directional change, defined by

$$D_{stst} = \frac{1}{N}\sum_{k\in P} b_k$$

where

$$b_k = \begin{cases} 1 & \text{if } (y_{t+1} - y_t)(\hat{y}_{t+1} - y_t) \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

is also used in our test to reflect the quality of the forecast in terms of directional changes.

Our aim is to select the neural network with the best performance on forecasting commodity future daily price movements over eight months, where the error of out-of-sample testing is the lowest compared to the other models. Table 4.1 and 4.2 contain the comparison of the out of sample test for the models respectively. In general, we are able to obtain at least 46% of accuracy for MLP model, 60% of accuracy for both NARX models and 78% of accuracy for LSTM model, while the optimal model can obtain at least 80% of forecasting accuracy for all eighteen products.

One would expect the NARX model to perform more stable at both the time and the frequency domain when compare to the traditional MLP model or LSTM model. This can be justified by the additional information supplied by the commodity futures in the same cluster.

When compare to short memory MLP model, all of the products exhibit the expected behaviors where all testing statistics show that NARX model performs better than MLP model. Since we inject more than one set of time series data for each of the NARX model in both the time and frequency domains from the clustering results, the forecasting model can take more market information with trends or characteristics into account. Based on the statistics, the additional information indeed bring better performances in estimation errors as well as directional changes.

However, our testing results indicate some additional interesting results. As we can see, the forecasting ability differs per product when we focus on the two NARX models clustered by time domain or frequency domain. In general, we can see that for error term, frequency domain has similar performances with time domain based on $NMSE$, $MAPE$ and R^2 statistics. The same consistency is not presented in corn, coca, aluminum, platinum, lean hogs and crude oil. In these products the time domain NARX performs relatively better than other models due to higher R^2 values. From the directional changes statistics, majority of the results show value higher than 0.5, which means the NARX models can predict the right direction of price movement at most of time. But the inconsistency of performances between two NARX models also applies to

Commodity	Measurement	Forecasting Model				
		ARIMA	MLP	NARX$_{Time}$	NARX$_{Freq}$	LSTM
Wheat	$NMSE$	0.0715	0.1811	0.1577	0.1474	0.1488
	$MAPE$	0.0164	0.0255	0.0243	0.0243	0.0237
	R^2	0.9308	0.8189	0.8423	**0.8526**	0.8512
	D_{stst}	0.5115	0.5345	0.5345	0.5287	0.5057
Soybean Oil	$NMSE$	0.0246	0.1216	0.0575	0.0507	0.0529
	$MAPE$	0.0070	0.0160	0.0109	0.0103	0.0104
	R^2	0.9762	0.8784	0.8784	**0.9493**	0.9471
	D_{stst}	0.5230	0.5747	0.5517	0.5805	0.5345
Soybean	$NMSE$	0.0218	0.0640	0.0494	0.0483	0.0444
	$MAPE$	0.0088	0.0165	0.0144	0.0143	0.0133
	R^2	0.9783	0.9360	0.9506	0.9517	**0.9556**
	D_{stst}	0.5345	0.5517	0.5230	0.5345	0.5517
Oats	$NMSE$	0.1152	0.3147	0.2144	0.2060	0.2173
	$MAPE$	0.0141	0.0259	0.0197	0.0197	0.0201
	R^2	0.8886	0.6853	0.7856	**0.7940**	0.7827
	D_{stst}	0.5057	0.5230	0.5690	0.5460	0.5230
Corn	$NMSE$	0.0585	0.2081	0.1055	0.1104	0.1026
	$MAPE$	0.0086	0.0177	0.0127	0.0125	0.0123
	R^2	0.9416	0.7919	0.8945	0.8896	**0.8974**
	D_{stst}	0.5747	0.4943	0.5747	0.5977	0.5575
Canola	$NMSE$	0.0579	0.2673	0.1566	0.1254	0.1800
	$MAPE$	0.0054	0.0128	0.0095	0.0085	0.0104
	R^2	0.9423	0.7327	0.8434	**0.8746**	0.8200
	D_{stst}	0.6379	0.5287	0.5345	0.4943	0.5345
Sugar	$NMSE$	0.0289	0.1644	0.0856	0.0742	0.0582
	$MAPE$	0.0126	0.0306	0.0235	0.0221	0.0188
	R^2	0.9722	0.8356	0.9144	0.9258	**0.9418**
	D_{stst}	0.5345	0.5747	0.4885	0.4885	0.5690
Orange juice	$NMSE$	0.0347	0.0980	0.0912	0.0778	0.0802
	$MAPE$	0.0102	0.0186	0.0187	0.0168	0.0165
	R^2	0.9660	0.9020	0.9088	**0.9222**	0.9198
	D_{stst}	0.5460	0.4828	0.4483	0.4425	0.5230
Cocoa	$NMSE$	0.0332	0.0720	0.0641	0.0756	0.0585
	$MAPE$	0.0162	0.0238	0.0226	0.0255	0.0221
	R^2	0.9676	0.9280	0.9359	0.9244	**0.9415**
	D_{stst}	0.5517	0.5632	0.5862	0.5747	0.5805
Coffee	$NMSE$	0.0502	0.2910	0.1100	0.1084	0.1667
	$MAPE$	0.0102	0.0262	0.0155	0.0153	0.0203
	R^2	0.9503	0.7090	0.8900	**0.8916**	0.8333
	D_{stst}	0.5460	0.5862	0.5575	0.5690	0.5230

Table 4.1: Performance Measurements for Different Machine Learning Forecasting Model

Commodity	Measurement	Forecasting Model				
		ARIMA	MLP	NARX$_{Time}$	NARX$_{Freq}$	LSTM
Aluminum	$NMSE$	0.1043	0.2100	0.3701	0.4025	0.2081
	$MAPE$	0.0119	0.0166	0.0226	0.0224	0.0162
	R^2	0.8998	0.7900	0.6299	0.5975	**0.7919**
	D_{stst}	0.5115	0.5172	0.5517	0.5460	0.5402
Gold	$NMSE$	0.0288	0.0549	0.1333	0.0764	0.0537
	$MAPE$	0.0046	0.0068	0.0116	0.0085	0.0066
	R^2	0.9714	0.9451	0.8667	0.9236	**0.9463**
	D_{stst}	0.5230	0.5747	0.5172	0.5460	0.5747
Copper	$NMSE$	0.0418	0.1227	0.0879	0.0795	0.0784
	$MAPE$	0.0089	0.0157	0.0127	0.0121	0.0119
	R^2	0.9600	0.8773	0.9121	0.9205	**0.9216**
	D_{stst}	0.5057	0.5172	0.5805	0.5575	0.5747
Palladium	$NMSE$	0.0785	0.1642	0.1423	0.1394	0.1707
	$MAPE$	0.0127	0.0195	0.0183	0.0182	0.0197
	R^2	0.9249	0.8358	0.8577	**0.8606**	0.8293
	D_{stst}	0.4713	0.5575	0.5517	0.5287	0.5000
Platinum	$NMSE$	0.0250	0.2426	0.0588	0.0626	0.0661
	$MAPE$	0.0089	0.0323	0.0139	0.0146	0.0148
	R^2	0.9750	0.7574	**0.9412**	0.9374	0.9339
	D_{stst}	0.5805	0.4770	0.5172	0.4828	0.5632
Lean hogs	$NMSE$	0.0479	0.2004	0.1025	0.1034	0.1073
	$MAPE$	0.0162	0.0445	0.0261	0.0263	0.0274
	R^2	0.9524	0.7996	**0.8975**	0.8966	0.8927
	D_{stst}	0.5632	0.4828	0.4828	0.4713	0.4655
Feeder cattle	$NMSE$	0.0727	0.5369	0.1949	0.1612	0.1631
	$MAPE$	0.0077	0.0263	0.0132	0.0119	0.0130
	R^2	0.9291	0.4631	0.8051	**0.8388**	0.8369
	D_{stst}	0.5287	0.4828	0.5690	0.5632	0.5057
Crude oil	$NMSE$	0.0830	0.2522	0.1635	0.1644	0.1933
	$MAPE$	0.0118	0.0225	0.0172	0.0173	0.0196
	R^2	0.9191	0.7478	**0.8365**	0.8356	0.8067
	D_{stst}	0.5747	0.4943	0.5517	0.5460	0.5230

Table 4.2: Performance Measurements for Different Machine Learning Forecasting Model

the directional changes, so we cannot conclude whether the time or frequency domain can generate overall better models as the statistics are close to each other.

Not surprisingly, LSTM has better or similar forecasting performance than all other models under investigated products. Both Figure 4.4 - 4.5 and Table 4.1 - 4.2 demonstrate the same conclusion. Since the LSTM algorithm models the underlying sequences with all its previous states, such a long memory model can capture more data characteristics. Another interesting observation from the best neural network model in terms of R^2 statistic, is it appears that the all the grain products and soft products have better performance with either LSTM or NARX models clustered by frequency domain other than time domain. One possible interpretation is that these two categories are less sensitive to time changes, so the forecasting results based on time domain cluster fail to outperform. But for livestocks and energy products, the LSTM models provide less accurate results compared to NARX model, which means the price of these two kinds of products relies more on other market information.

Based on the test results, it is clear that NARX forecasting model based on different clustering assumption generates performance better than the MLP method and is comparable to LSTM algorithm in both forecasting errors and directional changes. There is evidence of increased performance on the experiments that contain more relevant data as input. More specifically, clustering by time domain and frequency domain improves the model forecast-abilities by providing more correlated close-market data sequences to support the model trainings.

Chapter 5

Conclusions and Future Development

5.1 Conclusion and Contributions of the Thesis

Predicting financial times series using traditional models is a challenging task. In this research, our innovative data analysis framework provides an interesting perspective for clustering algorithms and commodity price forecasting applications. The research investigates the statistical characteristics of commodity data and the internal relationship among different commodity data based on functional clustering on time domain and frequency domain modeling. The aim of this thesis is to study the modeling methodologies and utilize machine learning techniques to combine financial time series clustering and forecasting models that outperforms traditional theories and existing models to inform better investment and risk management decisions.

The last chapter concludes the models, methodologies, algorithms and applications presented in each chapter and the contributions made. Finally, the summary of limitations of our current work and future suggestions are extended for further work.

Chapter 1: Introduction The first chapter in the thesis introduces the background and motivations of our research. Facing the two main challenges in the financial market, long memory in financial time series and dependence on economic variables, the lack of efficient model causes the difficulty to capture data

features and variation tendency for forecasting and risk management. The thesis structure and objectives are given to propose a new data analysis framework based on various machine learning algorithms to improve the efficiency of detecting optimal features in commodity data and the forecast performance. Theoretical background and literature on the Efficient Market Hypothesis (EMH) and time series model are reviewed. Given the weak form of the efficient market hypothesis, values from the remote part of a long memory time series can help forecast future returns. For the research on stylized facts, the chapter shows a description of the data used: daily index and futures data for eighteen different commodities from Bloomberg, along with the study of statistical characteristics with regard to four basic stylized facts: stationarity, dependence, skewness and kurtosis. We find that all of the commodity prices expect oats index appear to have unit roots with stationary increments, and all the products are long-range dependent, skewed and kurtotic with different fractional characteristics.

Chapter 2: Generalized Hurst Exponent The main purpose of the second chapter is to provide a possible alternative approach for the first challenge to model the long memory financial time series. We first introduce the long memory and ARFIMA model. Due to the fact that non-linear models are practically very difficult to implement, we consider ARIMA models and see how well they can capture the commodity prices statistical characteristics. Interestingly, unlike the R/S Hurst exponent, the generalized Hurst exponent of a linear process can be similar to the actual commodity daily data. We study statistical characteristics of an ARIMA$(p, 1, q)$ model with α-stable shocks, and observe that while this process has unit roots with skewed, kurtotic and stationary increments, its generalized Hurst exponent can be greater than $1/2$, which is consistent with the commodity daily index and future prices. According to our empirical observation, only wheat future index, wheat future, soybean oil future, oats future prices fail to have generalized Hurst exponent greater than $1/2$. As a result, we prove that a linear model still can be applied for modeling actual data. In addition, a strong implication of our discussions is that even though the generalized Hurst exponent is a measure of fractionality, it is not necessarily affected by a time series memory. Therefore, ARIMA$(p, 1, q)$ models with skewed and kurtotic (e.g., α-stable) shocks are suitable to model commodity daily index and future prices.

Chapter 3: Multidimensional Data Clustering The third chapter proposes one solution to analyze the dependence on high dimensional time series as mentioned in the second challenge. We start with the review of various functional clustering methods, then focus on two model-based functional data clustering methods from both time domain and frequency domain. For modeling multidimensional data, dimension reduction technique is performed to find principal variables in order to approximate the density function. On the time domain functional principal component analysis (FPCA) is adopted to build an optimal representation of curves into a function space of reduced dimension, from $\mathbb{R}^n$ to the L^2 space by determining principal components characterizing the variability of density f_t. On the frequency domain, the log-spectral density functions $\log\{f_i(\omega)\}$ is estimated by a linear combination of a common set of basis functions $\phi_k(\omega)$ and corresponding coefficients. Two experiments with commodity futures data are conducted for both methods. To select the optimal number of clusters, we consider the Elbow method with within-cluster sum of errors. Results show the comparison between two cluster methods based on time domain analysis and frequency domain analysis.

Chapter 4: Long Memory Time Series Forecasting The purpose of the fourth chapter is to perform multiple predictions for commodity future prices, which are long memory time series. First, the literature on Machine Learning applications and Artificial Neural Networks are introduced, following the basic supervised algorithms Backpropagation (BP) and Levenberg-Marquardt (LM) applied to the prediction of financial or commodity markets. In order to determine which inputs to be used in the model, the study applies the Recurrent Neural Network (RNN) that is based on a nonlinear autoregressive network with exogenous inputs (NARX) model that takes into account the clustering results, to obtain multiple-time-ahead predictions. The performance of the NARX model is compared with the performance of the traditional time series ARIMA model, classic machine learning multilayer perceptron (MLP) model and long short-term memory (LSTM) model which can capture long-range dependency. For performance evaluation, out-of-sample test through normalized mean square errors (NMSE), mean absolute percentage error (MAPE) and Directional Change test are used to measure the model performance of these three methodologies. The forecast results trained on a daily basis for future prices with respect to the last ten years. The result shows that our approach based on data clustering gives much better performance than the MLP model, and is

able to provide a high accuracy as LSTM approach for eight-month prediction horizon. Therefore, taking advantage of the interdependence structure between high-dimensional time series can improve the forecast accuracy.

5.2 Limitations and Future Development

In this section, we present some limitations of our work and provide potential ideas for future research.

One of the potential improvement of our research is on clustering methods. It is important that the model is robust when the data distribution is skewed, in order to provide the cluster results that converge for each experiment and have consistent economics findings. Hence prior to applying PCA, data transformation is applied so that the PCA can also be smooth. One way to approximate discrete data by a function is the roughness penalty approach. The roughness penalty can be introduced to ensure the desired smoothness on the FPC weight functions and apply the maximum penalized likelihood for parameter estimation, which can avoid data over-fitting and allow one to control the amount of smoothing by using a large number of basis functions. Once we can model curve approximation for data with skewed distribution, we can understand dynamics better and have more reliable clustering results for our future work.

Another limitation in our forecasting study is that only the commodity prices are considered as inputs in the NARX model. It would be interesting to introduce more economic factors to see if the networks can learn from the correlation between markets. However, we have not analyzed other multi-factors that are related to commodity price, such as temperature, weather, inflation, supply and demand in commodities, and other different asset classes, e.g. stocks and bonds, etc., as inputs to the network. Assuming these factors do exhibit different statistical properties and dependencies, as such it would be of interest to investigate the appropriate and informative inputs and potential use of advanced neural networks to relax the limit of multi-factor models in order to improve forecast performance. Another part of neural networks to be enhanced is the problem of overfitting the data, one of the major issues when the models are quite complicated. Therefore, regularization technique also needs to be introduced to our model and the optimal magnitude of the regularization coefficient remains to be explored.

Chapter 6

Appendix

In the following we present the proof of Corollary 4, tables of results and the tables of data descriptions.

6.1 Proof of Corollary 4

By (2.9) for $0 \leq N \leq i \leq k$ we have,

$$\begin{aligned}
\left|\sum_{j=0}^{k-1} c_{j+i-k+1}\right| &= \left|\sum_{j=k-1-i}^{k-1} c_{j+i-k+1}\right| = \left|\sum_{j=0}^{i} c_j\right| \\
&\geq \left|\sum_{j=0}^{N-1} c_j\right| - \left|\sum_{j=N}^{i} c_j\right| \\
&\geq \left|\sum_{j=0}^{N-1} c_j\right| - K_\rho \rho^N \frac{1-\rho^{i-N+1}}{1-\rho} \\
&\geq \left|\sum_{j=0}^{N-1} c_j\right| - \frac{K_\rho \rho^N}{1-\rho}.
\end{aligned} \tag{6.1}$$

On the other hand, since $|c_j| \leq M_\rho \rho^j, j = 1, 2, ...$, then $\sum_{j=0}^{\infty} c_j$ exists. We claim that $\sum_{j=0}^{\infty} c_j \neq 0$. To see this note first that since B has eigenvalues within the unite circle then $\sum_{i=0}^{\infty} B^i$ exists. Now, from (2.8) $z_t = \sum_{l=0}^{\infty} \left[B^l\right]_{1,1} \sum_{j=0}^{q} \theta_j \epsilon_{t-l-j}$,

which implies

$$\begin{aligned}\sum_{j=0}^{\infty} c_j &= \sum_{l=0}^{\infty} \left[B^l\right]_{1,1} \sum_{j=0}^{q} \theta_j \\ &= \sum_{j=0}^{q} \theta_j \sum_{l=0}^{\infty} \left[B^l\right]_{1,1} \\ &= \sum_{j=0}^{q} \theta_j \left[\sum_{l=0}^{\infty} B^l\right]_{1,1} \\ &= \sum_{j=0}^{q} \theta_j \left[(I-B)^{-1}\right]_{1,1} \\ &= \left[(I-B)^{-1}\right]_{1,1} \left(\sum_{j=0}^{q} \theta_j\right).\end{aligned}$$

Since by assumption $\left(\sum_{j=0}^{q} \theta_j\right) \neq 0$, we only need to show that $\left[(I-B)^{-1}\right]_{1,1} \neq 0$. Let us denote the first column of $(I-B)^{-1}$ by $\begin{bmatrix} h_2 \\ \vdots \\ h_p \end{bmatrix}$. In particular, $h_2 = \left[(I-B)^{-1}\right]_{1,1}$. Since $(I-B)(I-B)^{-1} = I$ we have

$$\begin{bmatrix} 1-b_2 & -b_3 & \cdots & -b_{p-1} & -b_p \\ -1 & 1 & \cdots & & 0 \\ 0 & -1 & \cdots & & 0 \\ \vdots & \vdots & \ddots & 1 & \vdots \\ 0 & 0 & \cdots & -1 & 1 \end{bmatrix} \begin{bmatrix} h_2 \\ \vdots \\ h_p \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

This implies that $h_2 - \sum_{i=2}^{p} h_i b_i = 1$ and $h_2 - h_3 = 0$,...,$h_{p-1} - h_p = 0$ which implies $h_2 = ... = h_p$ and $h_2\left(1 - \sum_{i=2}^{p} b_i\right) = 1$. The implication of this last relation then is that $h_2 \neq 0$ which means $\sum_{j=0}^{\infty} c_j \neq 0$. Knowing that $\sum_{j=0}^{\infty} c_j \neq 0$, there exists $\eta > 0$ such that for any large N_1, there exists $N > N_1$ so that $\left|\sum_{j=0}^{N-1} c_j\right| > \eta$. Now let N be large enough so that also $\frac{M_\rho \rho^N}{1-\rho} < \frac{\eta}{2}$. Combining this with (6.1) we obtain that for $0 \leq N \leq i \leq k$,$\left|\sum_{j=0}^{k-1} c_{j+i-k+1}\right| \geq \frac{\eta}{2}$. Now we

have the following

$$\begin{aligned}\sum_{i=0}^{\infty}\left|\sum_{j=0}^{k-1}c_{j+i-k+1}\right|^{\alpha} &= \sum_{i=0}^{N}\left|\sum_{j=0}^{k-1}c_{j+i-k+1}\right|^{\alpha} \\ &\quad+\sum_{i=N+1}^{k}\left|\sum_{j=0}^{k-1}c_{j+i-k+1}\right|^{\alpha} \\ &\quad+\sum_{i=k+1}^{\infty}\left|\sum_{j=0}^{k-1}c_{j+i-k+1}\right|^{\alpha} \\ &\geq 0+\sum_{i=N+1}^{k}\left(\frac{\eta}{2}\right)^{\alpha}+0=(k-N)\left(\frac{\eta}{2}\right)^{\alpha}.\end{aligned} \tag{6.2}$$

On the other hand, we have

$$\begin{aligned}\sum_{i=0}^{\infty}\left|\sum_{j=0}^{k-1}c_{j+i-k+1}\right|^{\alpha} &= \sum_{i=0}^{k-1}\left|\sum_{j=0}^{k-1}c_{j+i-k+1}\right|^{\alpha}+\sum_{i=k}^{\infty}\left|\sum_{j=0}^{k-1}c_{j+i-k+1}\right|^{\alpha} \\ &= \sum_{i=0}^{k-1}\left|\sum_{j=0}^{i}c_{j}\right|^{\alpha}+\sum_{i=k}^{\infty}\left|\sum_{j=0}^{k-1}c_{j+i+1}\right|^{\alpha} \\ &\leq \sum_{i=0}^{k-1}M_{\rho}^{\alpha}\left|\sum_{j=0}^{i}\rho^{j}\right|^{\alpha}+\sum_{i=k}^{\infty}M_{\rho}^{\alpha}\left|\sum_{j=0}^{k-1}\rho^{j+i+1}\right|^{\alpha} \\ &= M_{\rho}^{\alpha}\sum_{i=0}^{k-1}\left|\frac{1-\rho^{i+1}}{1-\rho}\right|^{\alpha}+M_{\rho}^{\alpha}\left|\frac{1-\rho^{k}}{1-\rho}\right|^{\alpha}\sum_{i=0}^{\infty}\rho^{\alpha(i+1)} \\ &\leq M_{\rho}^{\alpha}\sum_{i=0}^{k-1}\left|\frac{1-\rho^{k}}{1-\rho}\right|^{\alpha}+M_{\rho}^{\alpha}\left|\frac{1-\rho^{k}}{1-\rho}\right|^{\alpha}\frac{1}{1-\rho^{\alpha}} \\ &= M_{\rho}^{\alpha}\left(k+\frac{1}{1-\rho^{\alpha}}\right)\left|\frac{1-\rho^{k}}{1-\rho}\right|^{\alpha}.\end{aligned} \tag{6.3}$$

Now (6.2) and (6.3) imply

$$\frac{\log\left(\frac{(k-N-1)\left(\frac{\eta}{2}\right)^{\alpha}}{\sum_{i=0}^{\infty}|c_i|^{\alpha}}\right)}{\log(k)} \leq \frac{\log\left(\frac{\sum_{i=0}^{\infty}\left|\sum_{j=0}^{k-1}c_{j+i-k+1}\right|^{\alpha}}{\sum_{i=0}^{\infty}|c_{i-k+1}|^{\alpha}}\right)}{\log(k)} \leq \frac{\log\left(\frac{M_{\rho}^{\alpha}\left(k+\frac{1}{1-\rho^{\alpha}}\right)\left|\frac{1-\rho^{k}}{1-\rho}\right|^{\alpha}}{\sum_{i=0}^{\infty}|c_i|^{\alpha}}\right)}{\log(k)},$$

or

$$\frac{\log(k-N-1)+\log\left(\frac{\left(\frac{\eta}{2}\right)^{\alpha}}{\sum_{i=0}^{\infty}|c_i|^{\alpha}}\right)}{\log(k)}$$
$$\leq \frac{\log\left(\frac{\sum_{i=0}^{\infty}\left|\sum_{j=0}^{k-1}c_{j+i-k+1}\right|^{\alpha}}{\sum_{i=0}^{\infty}|c_{i-k+1}|^{\alpha}}\right)}{\log(k)}$$
$$\leq \frac{\log\left(k+\frac{1}{1-\rho^{\alpha}}\right)+\log\left(\frac{M_{\rho}^{\alpha}\left|\frac{1-\rho^{k}}{1-\rho}\right|^{\alpha}}{\sum_{i=0}^{\infty}|c_i|^{\alpha}}\right)}{\log(k)}.$$

By sending $k\to\infty$ we get we get

$$\lim_{k\to\infty}\frac{\log\left(\frac{\sum_{i=0}^{\infty}\left|\sum_{j=0}^{k-1}c_{j+i-k+1}\right|^{\alpha}}{\sum_{i=0}^{\infty}|c_{i-k+1}|^{\alpha}}\right)}{\log(k)}=1.$$

6.2 Tables of Results

Commodity	Description	Period
	Grains	
Wheat	Bloomberg Generic 1ST W Wheat CBOT	1971/12/09 - 2018/08/31
	No.2 soft red winter wheat IL index	1992/01/02 - 2018/08/31
Soybean Oil	Bloomberg Generic 1ST BO Soybean oil CBOT	1972/12/23 - 2018/08/31
	Crude soybean oil spot IL index	1984/01/02 - 2018/08/31
Soybean	Bloomberg Generic 1ST S Soybean CBOT	1970/01/02 - 2018/08/31
	Yellow beans FOB Chicago Index	1996/01/02 - 2018/08/31
Oats	Bloomberg Generic 1ST O Oats CBOT	1970/01/02 - 2018/08/31
	Kansas Oats Index (OATAUSKS)	2008/06/19 - 2018/08/31
Corn	Bloomberg Generic 1ST C Corn CBOT	1970/06/26 - 2018/08/31
	Illinois NC Spot Price Index (CORNILNC)	1992/01/02 - 2018/08/31
Canola	Bloomberg Generic 1ST WC Canola WCE	1982/01/04 - 2018/08/31
	Canola Spot Canada Vancouver Index (WCEFC1VA)	2003/01/02 - 2018/08/31
	Softs	
Sugar	Bloomberg Generic 1ST SB Sugar NYBOT	1970/06/26 - 2018/08/31
	CSCE No.11 Sugar Spot Golable Index	1990/12/03 - 2018/08/31
Orange juice	Bloomberg Generic 1ST JO NYBOT	1970/06/26 - 2018/08/31
	Orange juice spot prices index	1994/01/03 - 2018/08/31
Cocoa	Bloomberg Generic 1ST CC Cocoa NYBOT	1970/06/26 - 2018/08/31
	Ivry Grd1 Coco Bean Spot Index	1997/01/02 - 2018/08/31
Coffee	Bloomberg Generic 1ST KC Coffee NYBOT	1972/08/26 - 2018/08/31
	Colombia Milds New York Index	1997/07/01 - 2018/08/31
	Metals	
Aluminum	Bloomberg Generic 1ST LA future NYMEX	1997/07/23 - 2018/08/31
	S&P 500 Aluminum index	1989/09/11 - 2018/08/31
Gold	Bloomberg Generic 1ST GC future NYMEX	1975/01/01 - 2018/08/31
	NYSE Arca Gold BUGS Index	1994/12/26 - 2018/08/31
Copper	Bloomberg 3M copper future LME	1986/04/01 - 2018/08/31
	Bloomberg S&P GSCI copper index CME	1977/01/07 - 2018/08/31
Palladium	Bloomberg Generic 1ST PA future NYMEX	1986/04/14 - 2018/08/31
	Bloomberg S&P GSCI palladium index	2008/09/10 - 2018/08/31
Platinum	Bloomberg Generic 1ST PL future NYMEX	1986/04/30 - 2018/08/31
	Bloomberg platinum index LME $OZ currency	1992/01/02 - 2018/08/31
	Livestock and Meats	
Lean hogs	Bloomberg Generic 1ST LH lean hogs CME	1986/04/01 - 2018/08/31
	S&P GSCI lean hogs spot index	1976/01/07 - 2018/08/31
Feeder cattle	Bloomberg Generic 1ST FC cattle feeder CME	1989/11/08 - 2018/08/31
	Bloomberg feeder cattle index CME	1996/01/22 - 2018/08/31
	Energy	
Crude oil	Bloomberg Generic 1ST CL crude oil NYMEX	1989/03/08 - 2018/08/31
	Bloomberg Brent crude oil index ICE	1998/06/15 - 2018/08/31

Table 6.1: Commodity Data Description

Commodity	Lags	ADF_P	$KPSS_P$	ADF_P	$KPSS_P$	R/S	Skew.	Kurt.
		log prices		Increments		Hurst		
Wheat Future	1	0.798	0.01	0.001	0.1	0.530	-0.8318	21.4787
	10	0.807	0.01	0.001	0.1			
	50	0.814	0.01	0.001	0.1			
Wheat Index	1	0.498	0.01	0.001	0.1	0.496	-0.6528	15.3956
	10	0.494	0.01	0.001	0.1			
	50	0.552	0.01	0.001	0.1			
Soybean Oil Future	1	0.507	0.01	0.001	0.1	0.548	-0.0489	5.5908
	10	0.528	0.01	0.001	0.1			
	50	0.573	0.01	0.001	0.1			
Soybean Oil Index	1	0.584	0.01	0.001	0.1	0.548	0.0203	44.6239
	10	0.578	0.01	0.001	0.1			
	50	0.56	0.01	0.001	0.1			
Soybean Future	1	0.838	0.01	0.001	0.1	0.542	-0.6364	9.5364
	10	0.838	0.01	0.001	0.1			
	50	0.802	0.01	0.001	0.1			
Soybean Index	1	0.637	0.01	0.001	0.1	0.566	-0.3591	9.4468
	10	0.632	0.01	0.001	0.1			
	50	0.604	0.01	0.001	0.1			
Oats Future	1	0.759	0.01	0.001	0.1	0.499	-1.0821	16.2376
	10	0.768	0.01	0.001	0.1			
	50	0.809	0.01	0.001	0.1			
Oats Index	1	0.397	0.01	0.001	0.1	0.438	-0.2052	17.7400
	10	0.351	0.01	0.001	**0.01**			
	50	0.373	0.01	0.001	**0.01**			
Corn Future	1	0.804	0.01	0.001	0.1	0.535	-1.0222	53.5188
	10	0.803	0.01	0.001	0.1			
	50	0.745	0.01	0.001	0.1			
Corn Index	1	0.512	0.01	0.001	0.1	0.555	-0.1560	8.2449
	10	0.482	0.01	0.001	0.1			
	50	0.417	0.01	0.001	0.1			
Canola Future	1	0.763	0.01	0.001	0.1	0.552	-0.9982	18.4464
	10	0.752	0.01	0.001	0.1			
	50	0.736	0.01	0.001	0.1			
Canola Index	1	0.699	0.01	0.001	0.1	0.559	-3.0055	269.5416
	10	0.728	0.01	0.001	0.1			
	50	0.747	0.01	0.001	0.1			

Table 6.2: Unit root test, increments stationary test, R/S Hurst exponent, skewness and kurtosis for stylized facts of wheat, soybean oil, soybean, oats, corn, canola.

Commodity	Lags	ADF$_P$	KPSS$_P$	ADF$_P$	KPSS$_P$	R/S	Skew.	Kurt.
		log prices		Increments		Hurst		
Sugar Future	1	0.552	0.01	0.001	0.1	0.548	0.1804	11.1910
	10	0.564	0.01	0.001	0.1			
	50	0.523	0.01	0.001	0.1			
Sugar Index	1	0.864	0.01	0.001	0.1	0.544	-0.3139	6.6039
	10	0.872	0.01	0.001	0.1			
	50	0.873	0.01	0.001	0.1			
Orange Juice Future	1	0.755	0.01	0.001	0.1	0.518	0.4386	14.7120
	10	0.749	0.01	0.001	0.1			
	50	0.765	0.01	0.001	0.1			
Orange Juice Index	1	0.646	0.01	0.001	0.1	0.54	1.0189	13.5505
	10	0.668	0.01	0.001	0.1			
	50	0.692	0.015	0.001	0.1			
Cocoa Future	1	0.861	0.01	0.001	0.1	0.54	-0.1577	6.1614
	10	0.866	0.01	0.001	0.1			
	50	0.852	0.01	0.001	0.1			
Cocoa Index	1	0.857	0.01	0.001	0.1	0.56	-0.1862	6.2647
	10	0.859	0.01	0.001	0.1			
	50	0.891	0.01	0.001	0.1			
Coffee Future	1	0.669	0.01	0.001	0.1	0.539	-0.0548	12.3571
	10	0.671	0.01	0.001	0.1			
	50	0.659	0.01	0.001	0.1			
Coffee Index	1	0.542	0.01	0.001	0.1	0.547	0.2620	8.2271
	10	0.562	0.01	0.001	0.093			
	50	0.508	0.01	0.001	0.062			
Aluminum Future	1	0.633	0.01	0.001	0.1	0.562	-0.32	7.08
	10	0.611	0.01	0.001	0.1			
	50	0.636	0.01	0.001	0.1			
Aluminum Index	1	0.501	0.01	0.001	0.1	0.643	-0.0972	11.8983
	10	0.518	0.01	0.001	0.1			
	50	0.533	0.01	0.001	0.1			
Gold Future	1	0.964	0.01	0.001	0.1	0.589	-0.0933	10.1068
	10	0.962	0.01	0.001	0.1			
	50	0.976	0.01	0.001	0.1			
Gold Index	1	0.99	0.01	0.001	**0.01**	0.58	0.3562	8.5216
	10	0.974	0.01	0.001	**0.01**			
	50	0.969	0.01	0.001	**0.019**			

Table 6.3: Unit root test, increments stationary test, R/S Hurst exponent, skewness and kurtosis for stylized facts of sugar, orange juice, cocoa, coffee, aluminum, gold.

Commodity	Lags	ADF_P	$KPSS_P$	ADF_P	$KPSS_P$	R/S	Skew.	Kurt.
		Log prices		Increments		Hurst		
Copper Future	1	0.902	0.01	0.001	0.1	0.596	-0.3530	9.1628
	10	0.891	0.01	0.001	0.1			
	50	0.862	0.01	0.001	0.1			
Copper Index	1	0.862	0.01	0.001	0.1	0.553	-0.2225	6.9661
	10	0.854	0.01	0.001	0.1			
	50	0.838	0.01	0.001	0.1			
Palladium Future	1	0.881	0.01	0.001	0.1	0.608	-0.1757	8.7960
	10	0.887	0.01	0.001	0.1			
	50	0.854	0.01	0.001	0.1			
Palladium Index	1	0.889	0.01	0.001	0.1	0.55	-0.5176	6.5479
	10	0.913	0.01	0.001	0.1			
	50	0.941	0.01	0.001	0.1			
Platinum Future	1	0.999	0.01	0.001	0.1	0.55	-1.2066	24.0771
	10	0.999	0.01	0.071	0.1			
	50	0.999	0.01	0.369	0.1			
Platinum Index	1	0.898	0.01	0.001	0.1	0.568	-0.5388	12.8654
	10	0.901	0.01	0.001	0.1			
	50	0.86	0.01	0.001	0.1			
Lean Hogs Future	1	0.677	0.01	0.001	0.1	0.551	-0.1201	39.0743
	10	0.681	0.01	0.001	0.1			
	50	0.706	0.01	0.001	0.1			
Lean Hogs Index	1	0.575	0.01	0.001	0.1	0.54	-0.0267	3.8566
	10	0.59	0.01	0.001	0.1			
	50	0.564	0.01	0.001	0.1			
Feeder Cattle Future	1	0.956	0.01	0.001	0.1	0.582	0.0657	16.8000
	10	0.963	0.01	0.001	0.1			
	50	0.938	0.01	0.001	0.1			
Feeder Cattle Index	1	0.997	0.01	0.001	0.079	0.596	-1.1218	26.1202
	10	0.991	0.01	0.001	0.1			
	50	0.975	0.01	0.001	0.1			
Crude Oil Future	1	0.733	0.01	0.001	0.1	0.555	-0.8582	19.2292
	10	0.767	0.01	0.001	0.1			
	50	0.689	0.01	0.001	0.1			
Crude Oil Index	1	0.859	0.01	0.001	0.1	0.597	-0.3301	5.7561
	10	0.846	0.01	0.001	0.1			
	50	0.811	0.01	0.001	0.1			

Table 6.4: Unit root test, increments stationary test, R/S Hurst exponent, skewness and kurtosis for stylized facts of copper, palladium, platinum, lean hogs, feeder cattle, crude oil.

Commodity	Generalized Hurst Exponent			α	Commodity	Generalized Hurst Exponent			α
	Mean	5%	95%			Mean	5%	95%	
Wheat Future	**0.492**	0.488	0.500	2.031	Wheat Index	0.504	0.503	0.505	1.986
Soybean Oil Future	0.514	0.512	0.517	1.944	Soybean Oil Index	0.537	0.534	0.539	1.861
Soybean Future	**0.496**	0.494	0.497	2.014	Soybean Index	0.511	0.509	0.512	1.958
Oats Future	**0.492**	0.491	0.492	2.034	Oats Index	0.524	0.513	0.530	1.908
Corn Future	0.538	0.535	0.539	1.859	Corn Index	0.532	0.531	0.535	1.878
Canola Future	0.529	0.527	0.530	1.891	Canola Index	0.546	0.544	0.547	1.832
Sugar Future	0.502	0.498	0.506	1.991	Sugar Index	0.502	0.501	0.502	1.994
Orange Juice Future	0.532	0.532	0.533	1.878	Orange Juice Index	0.501	0.490	0.513	1.997
Cocoa Future	**0.488**	0.485	0.489	2.049	Cocoa Index	**0.476**	0.466	0.481	2.101
Coffee Future	0.514	0.513	0.515	1.944	Coffee Index	0.516	0.513	0.520	1.937
Aluminum Future	**0.496**	0.494	0.498	2.015	Aluminum Index	0.596	0.589	0.602	1.677
Gold Future	0.550	0.548	0.552	1.819	Gold Index	**0.487**	0.485	0.489	2.055
Copper Future	0.529	0.526	0.534	1.889	Copper Index	0.519	0.516	0.520	1.928
Palladium Future	0.563	0.560	0.565	1.777	Palladium Index	**0.464**	0.405	0.520	2.157
Platinum Future	0.510	0.507	0.514	1.959	Platinum Index	0.530	0.528	0.532	1.888
Lean Hogs Future	0.559	0.552	0.567	1.789	Lean Hogs Index	**0.477**	0.475	0.478	2.098
Feeder Cattle Future	0.564	0.562	0.566	1.773	Feeder Cattle Index	0.663	0.661	0.666	1.508
Crude Oil Future	0.508	0.505	0.511	1.969	Crude Oil Index	0.552	0.548	0.556	1.813

Table 6.5: Expected values, 5% and 95% quantile values of Generalized Hurst exponent, and theoretical expected values of α.

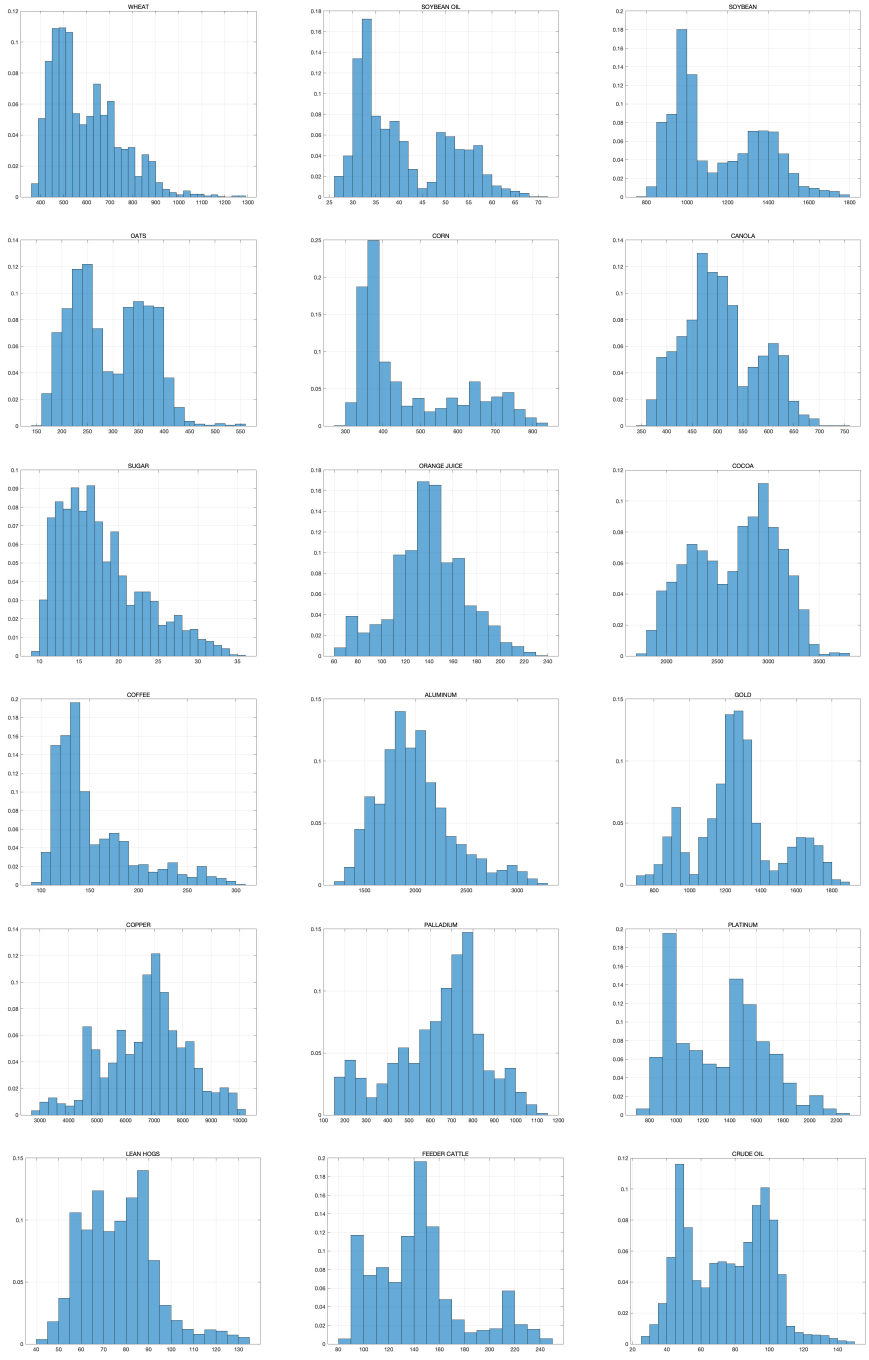

Figure 6.1: Histogram Plot for Commodity Future Price

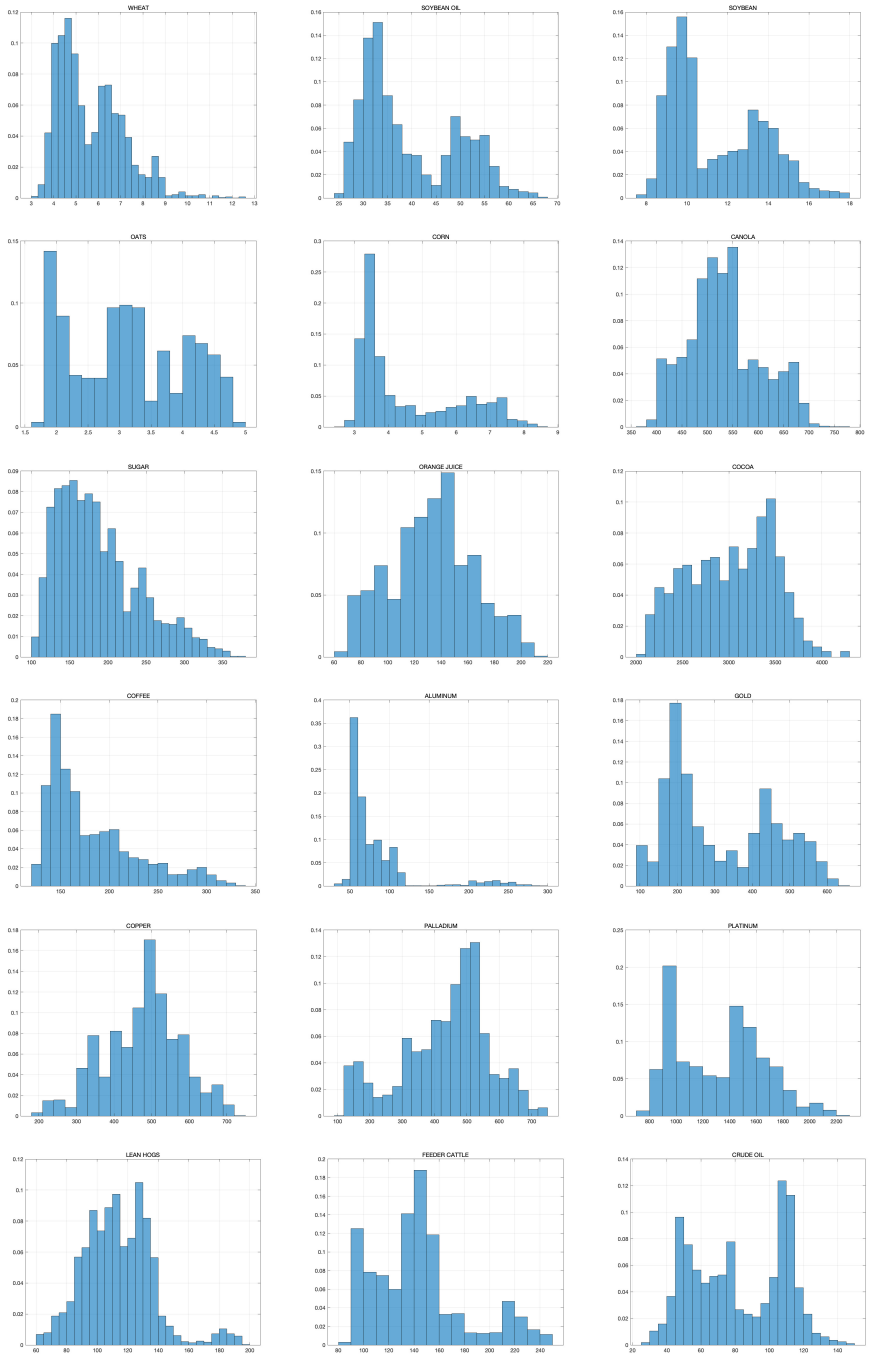

Figure 6.2: Histogram Plot for Commodity Index Price

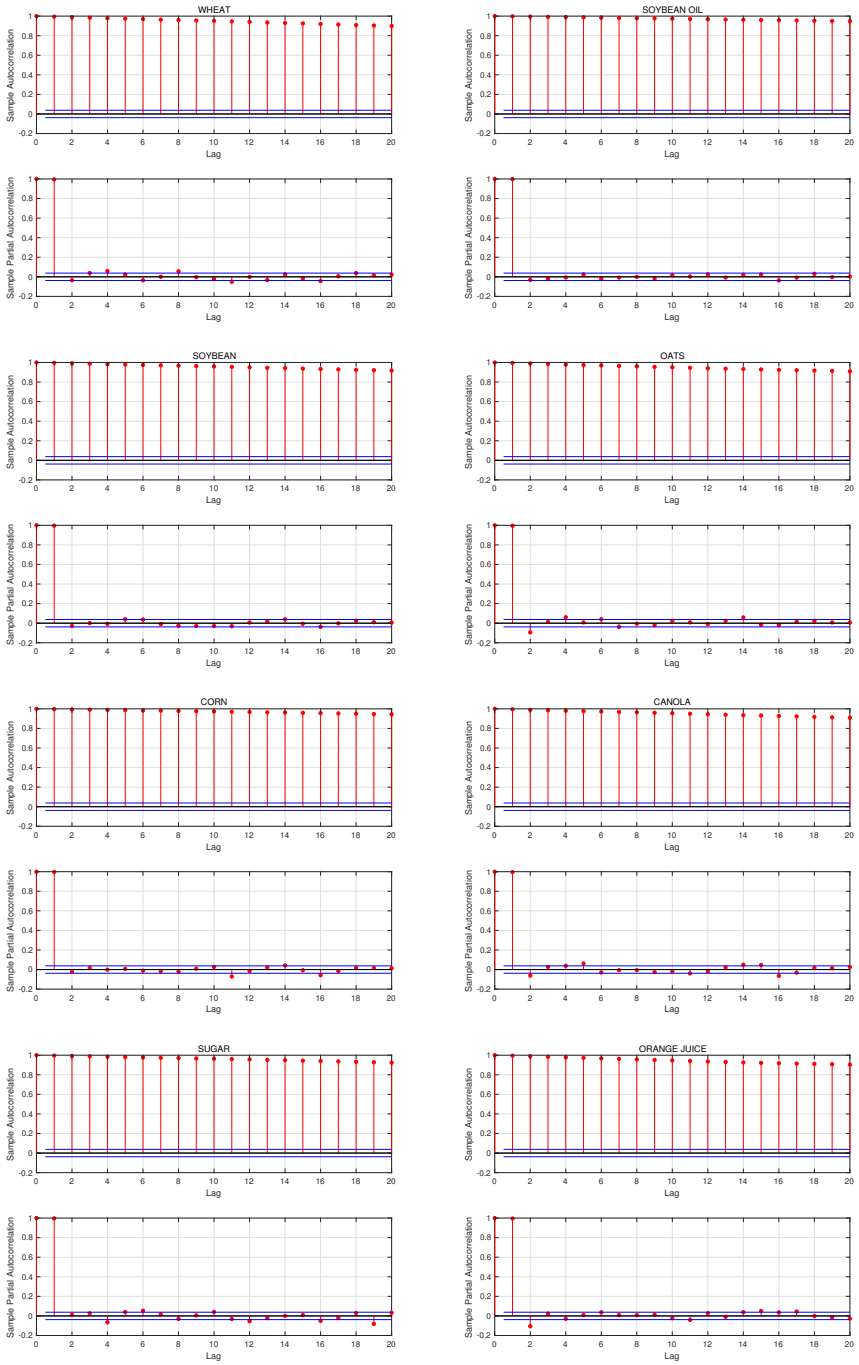

Figure 6.3: Sample ACF and PACF for Community Future Price of wheat, soybean oil, soybean, oats, corn, canola, sugar, orange juice.

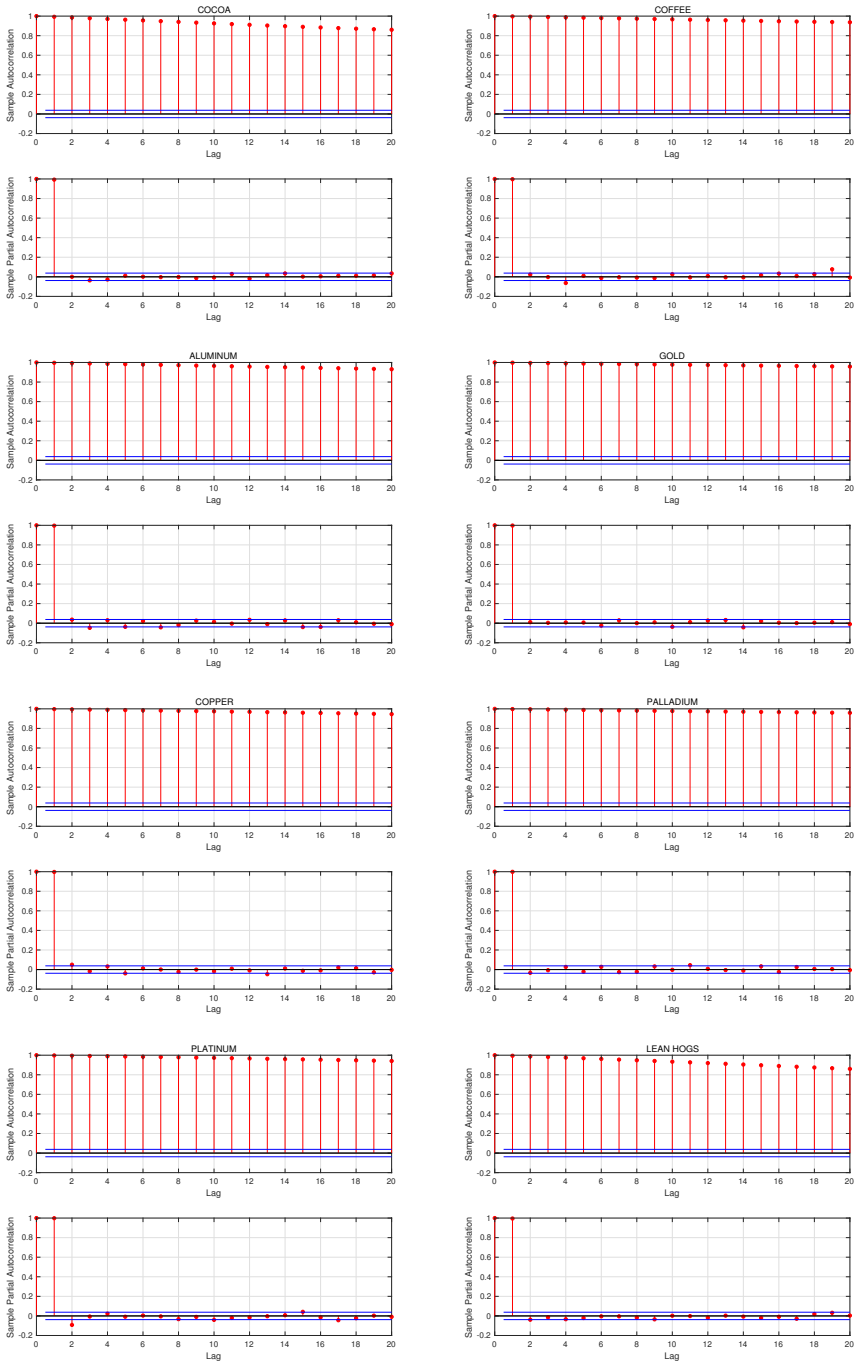

Figure 6.4: Sample ACF and PACF for Community Future Price of cocoa, coffee, aluminum, gold, copper, palladium, platinum, lean hogs.

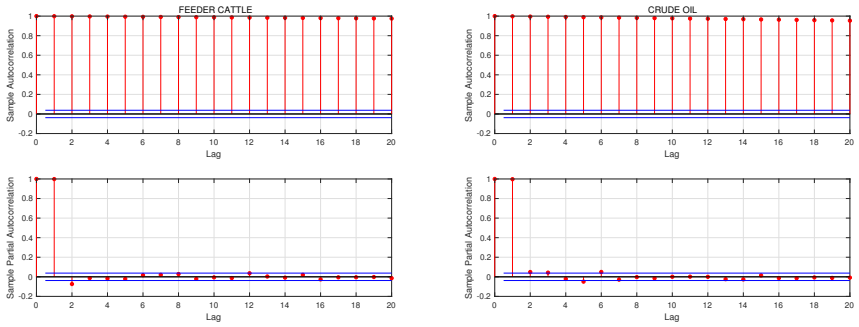

Figure 6.5: Sample ACF and PACF for Community Future Price of feeder cattle, crude oil.

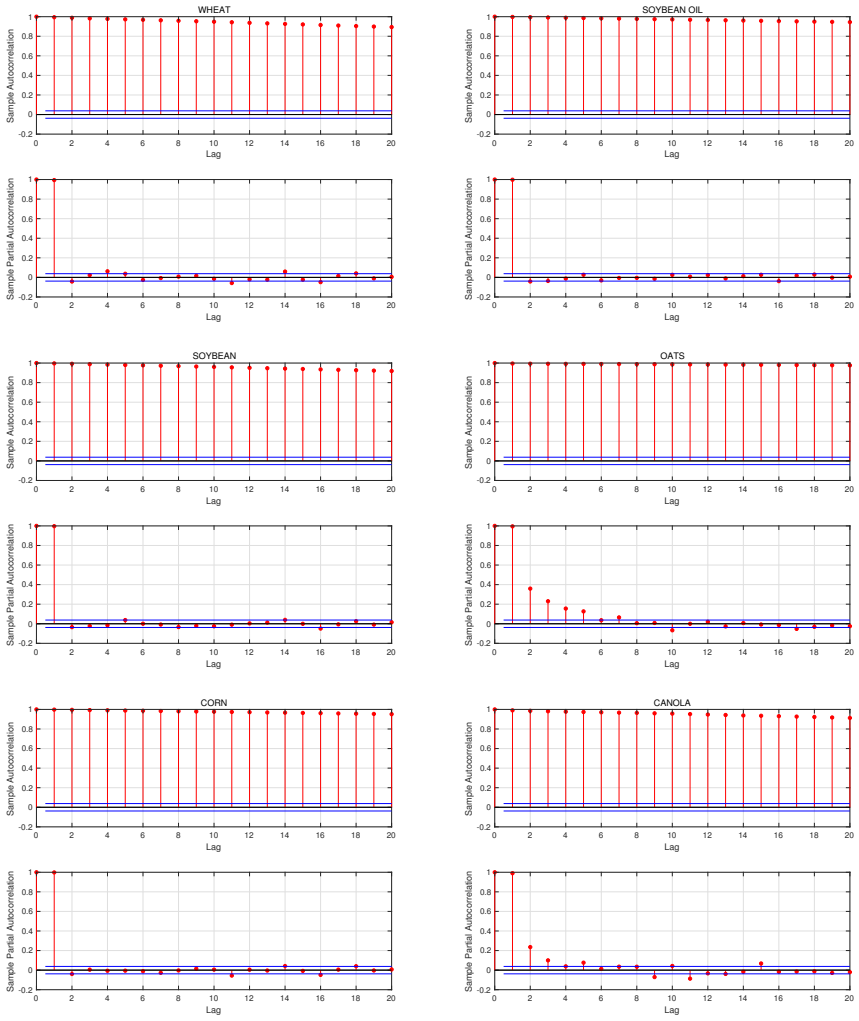

Figure 6.6: Sample ACF and PACF for Community Index Price of wheat, soybean oil, soybean, oats, corn, canola.

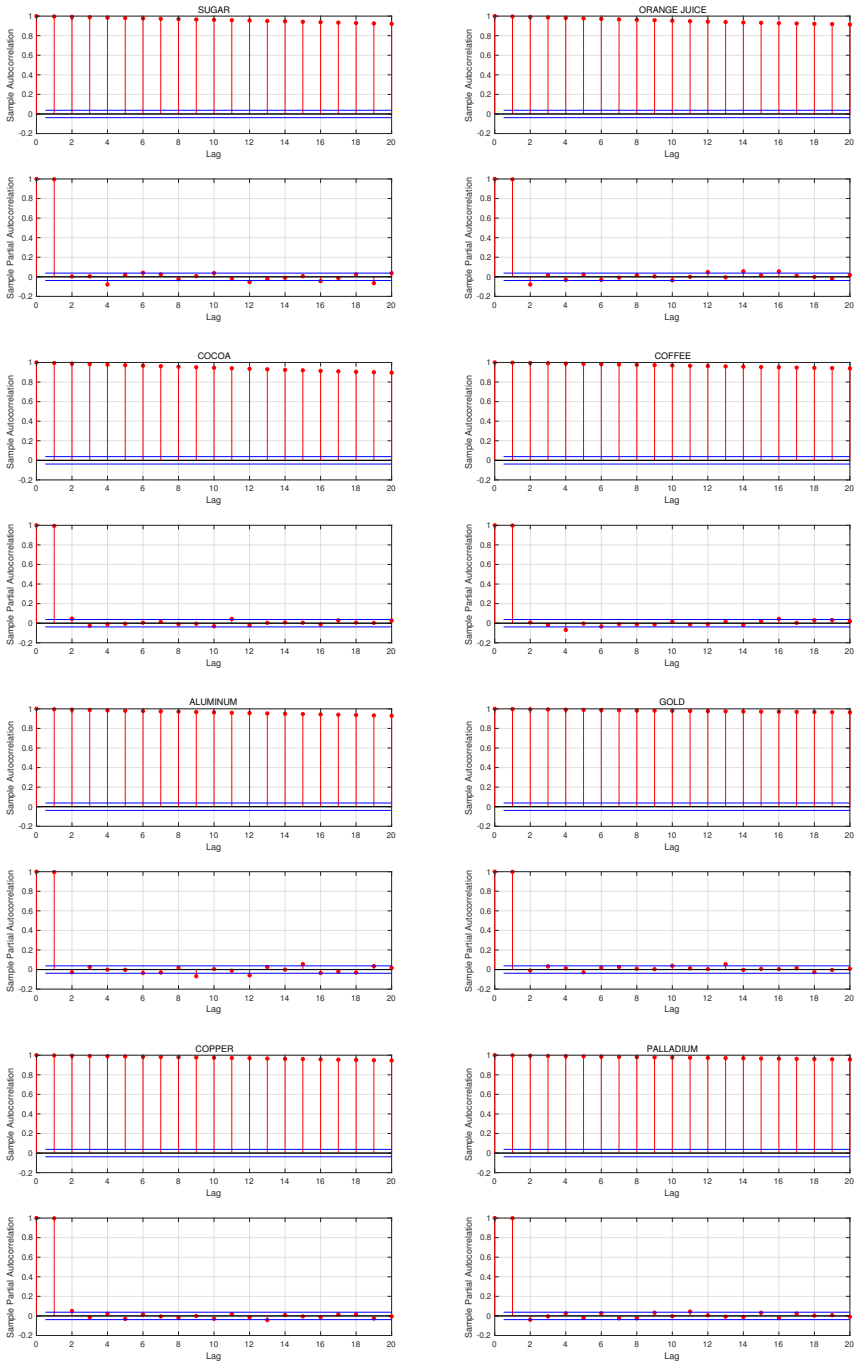

Figure 6.7: Sample ACF and PACF for Community Index Price of sugar, orange juice, cocoa, coffee, aluminum, gold, copper, palladium.

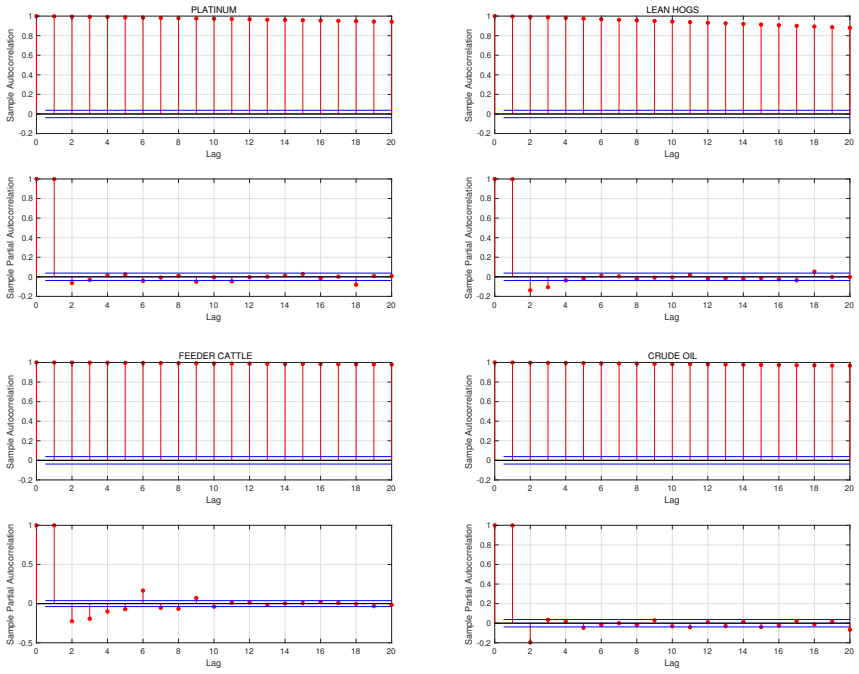

Figure 6.8: Sample ACF and PACF for Community Index Price of platinum; lean hogs, feeder cattle, crude oil.

GHE	α=0.1	α=0.2	α=0.3	α=0.4	α=0.5	α=0.6	α=0.7	α=0.8	α=0.9	α=1.0
k=30	1.6674	1.6636	1.6575	1.6356	1.5707	1.4535	1.3398	1.2069	1.0977	0.9899
k=40	1.6673	1.6630	1.6555	1.6322	1.5664	1.4479	1.3366	1.2041	1.0958	0.9885
k=50	1.6666	1.6613	1.6536	1.6292	1.5630	1.4428	1.3341	1.2017	1.0944	0.9871
k=60	1.6661	1.6602	1.6521	1.6268	1.5596	1.4383	1.3318	1.1995	1.0935	0.9857
k=70	1.6660	1.6593	1.6508	1.6248	1.5565	1.4344	1.3296	1.1975	1.0929	0.9845
k=80	1.6660	1.6587	1.6493	1.6229	1.5534	1.4311	1.3276	1.1957	1.0922	0.9833
k=90	1.6661	1.6581	1.6480	1.6213	1.5506	1.4283	1.3258	1.1942	1.0914	0.9819
k=100	1.6662	1.6575	1.6468	1.6199	1.5483	1.4259	1.3241	1.1927	1.0907	0.9807
k=110	1.6664	1.6568	1.6455	1.6185	1.5460	1.4236	1.3226	1.1913	1.0901	0.9796
k=120	1.6663	1.6563	1.6445	1.6172	1.5437	1.4214	1.3213	1.1899	1.0895	0.9788
k=130	1.6663	1.6558	1.6435	1.6160	1.5416	1.4194	1.3200	1.1885	1.0889	0.9782
k=140	1.6662	1.6552	1.6426	1.6148	1.5398	1.4175	1.3187	1.1871	1.0881	0.9778
k=150	1.6661	1.6546	1.6416	1.6137	1.5381	1.4156	1.3175	1.1859	1.0875	0.9775
k=160	1.6659	1.6541	1.6408	1.6125	1.5364	1.4137	1.3163	1.1848	1.0869	0.9771
k=170	1.6658	1.6535	1.6401	1.6114	1.5347	1.4120	1.3151	1.1837	1.0863	0.9767
k=180	1.6657	1.6530	1.6395	1.6103	1.5331	1.4104	1.3139	1.1828	1.0858	0.9762
k=190	1.6655	1.6525	1.6387	1.6092	1.5316	1.4089	1.3129	1.1818	1.0853	0.9758
k=200	1.6654	1.6520	1.6380	1.6082	1.5302	1.4075	1.3118	1.1810	1.0848	0.9754
k=210	1.6652	1.6514	1.6372	1.6074	1.5288	1.4062	1.3109	1.1802	1.0843	0.9750
k=220	1.6648	1.6507	1.6362	1.6066	1.5275	1.4049	1.3100	1.1795	1.0838	0.9745
k=230	1.6643	1.6501	1.6352	1.6058	1.5262	1.4037	1.3091	1.1787	1.0833	0.9740
k=240	1.6639	1.6496	1.6344	1.6050	1.5250	1.4025	1.3081	1.1780	1.0828	0.9735
k=250	1.6634	1.6491	1.6334	1.6043	1.5238	1.4013	1.3071	1.1773	1.0824	0.9730
k=260	1.6630	1.6487	1.6324	1.6036	1.5228	1.4002	1.3061	1.1767	1.0821	0.9727
k=270	1.6626	1.6483	1.6315	1.6030	1.5217	1.3991	1.3051	1.1761	1.0819	0.9724
k=280	1.6623	1.6479	1.6307	1.6024	1.5207	1.3979	1.3043	1.1754	1.0817	0.9722
k=290	1.6620	1.6476	1.6300	1.6018	1.5197	1.3968	1.3034	1.1748	1.0814	0.9720
k=300	1.6617	1.6472	1.6292	1.6012	1.5188	1.3957	1.3025	1.1742	1.0812	0.9717
k=310	1.6615	1.6470	1.6285	1.6007	1.5178	1.3946	1.3016	1.1736	1.0809	0.9713
k=320	1.6613	1.6467	1.6279	1.6001	1.5168	1.3935	1.3007	1.1730	1.0807	0.9710
k=330	1.6612	1.6464	1.6273	1.5994	1.5158	1.3924	1.2997	1.1725	1.0804	0.9707
k=340	1.6610	1.6461	1.6266	1.5988	1.5149	1.3914	1.2988	1.1720	1.0802	0.9704
k=350	1.6609	1.6458	1.6260	1.5982	1.5140	1.3903	1.2979	1.1715	1.0799	0.9701

Table 6.6: Expected values of generalized Hurst exponent of simulated 100 time series for different α in the columns, and different simulated time series lengths k in the rows.

GHE	α=1.1	α=1.2	α=1.3	α=1.4	α=1.5	α=1.6	α=1.7	α=1.8	α=1.9	α=2.0
k=30	0.9002	0.8249	0.7607	0.7076	0.6631	0.6250	0.5888	0.5588	0.5264	0.4999
k=40	0.8996	0.8240	0.7596	0.7063	0.6622	0.6248	0.5888	0.5593	0.5263	0.4999
k=50	0.8989	0.8227	0.7586	0.7054	0.6610	0.6246	0.5884	0.5594	0.5258	0.5000
k=60	0.8985	0.8213	0.7582	0.7051	0.6600	0.6247	0.5883	0.5600	0.5258	0.4995
k=70	0.8980	0.8200	0.7581	0.7050	0.6591	0.6246	0.5882	0.5602	0.5260	0.4994
k=80	0.8974	0.8188	0.7576	0.7045	0.6579	0.6246	0.5884	0.5604	0.5259	0.4992
k=90	0.8966	0.8179	0.7569	0.7041	0.6567	0.6245	0.5884	0.5606	0.5258	0.4986
k=100	0.8958	0.8172	0.7566	0.7038	0.6558	0.6245	0.5886	0.5607	0.5257	0.4983
k=110	0.8952	0.8165	0.7564	0.7032	0.6550	0.6248	0.5890	0.5607	0.5256	0.4985
k=120	0.8947	0.8159	0.7561	0.7027	0.6544	0.6249	0.5895	0.5610	0.5256	0.4983
k=130	0.8944	0.8153	0.7559	0.7021	0.6538	0.6250	0.5897	0.5610	0.5253	0.4982
k=140	0.8941	0.8148	0.7554	0.7015	0.6534	0.6248	0.5899	0.5610	0.5254	0.4981
k=150	0.8937	0.8143	0.7551	0.7010	0.6533	0.6247	0.5900	0.5610	0.5253	0.4978
k=160	0.8934	0.8140	0.7548	0.7004	0.6530	0.6245	0.5899	0.5612	0.5252	0.4975
k=170	0.8931	0.8136	0.7544	0.6998	0.6527	0.6244	0.5897	0.5613	0.5252	0.4976
k=180	0.8928	0.8131	0.7538	0.6993	0.6524	0.6243	0.5894	0.5612	0.5252	0.4977
k=190	0.8924	0.8126	0.7534	0.6989	0.6521	0.6244	0.5891	0.5612	0.5252	0.4979
k=200	0.8919	0.8122	0.7530	0.6986	0.6519	0.6246	0.5888	0.5613	0.5252	0.4979
k=210	0.8913	0.8117	0.7525	0.6983	0.6518	0.6247	0.5884	0.5614	0.5252	0.4980
k=220	0.8908	0.8112	0.7521	0.6979	0.6515	0.6245	0.5882	0.5614	0.5251	0.4978
k=230	0.8904	0.8108	0.7516	0.6976	0.6513	0.6242	0.5879	0.5617	0.5248	0.4976
k=240	0.8899	0.8104	0.7511	0.6972	0.6511	0.6239	0.5876	0.5621	0.5246	0.4977
k=250	0.8894	0.8100	0.7508	0.6970	0.6510	0.6237	0.5875	0.5624	0.5246	0.4978
k=260	0.8890	0.8096	0.7504	0.6968	0.6508	0.6236	0.5874	0.5626	0.5244	0.4979
k=270	0.8886	0.8092	0.7501	0.6964	0.6507	0.6235	0.5871	0.5626	0.5243	0.4980
k=280	0.8881	0.8088	0.7497	0.6961	0.6507	0.6235	0.5870	0.5627	0.5244	0.4979
k=290	0.8877	0.8084	0.7494	0.6957	0.6508	0.6234	0.5868	0.5629	0.5245	0.4980
k=300	0.8873	0.8081	0.7491	0.6953	0.6507	0.6235	0.5867	0.5630	0.5245	0.4979
k=310	0.8870	0.8078	0.7488	0.6949	0.6506	0.6235	0.5866	0.5632	0.5246	0.4978
k=320	0.8865	0.8074	0.7486	0.6946	0.6505	0.6235	0.5865	0.5635	0.5245	0.4977
k=330	0.8861	0.8072	0.7482	0.6942	0.6503	0.6235	0.5864	0.5638	0.5244	0.4976
k=340	0.8857	0.8069	0.7478	0.6938	0.6501	0.6235	0.5864	0.5639	0.5244	0.4977
k=350	0.8853	0.8065	0.7475	0.6935	0.6500	0.6235	0.5864	0.5641	0.5243	0.4978

Table 6.7: Expected values of generalized Hurst exponent of simulated 100 time series for different α in the columns, and different simulated time series lengths k in the rows.

Commodity	Maximum Log-likelihood		Estimated α
	α-stable	normal	
Wheat Future	-1.5155	-1.4189	1.8868
Wheat Index	-1.4991	-1.4189	2.0163
Soybean Oil Future	-1.5159	-1.4189	1.8248
Soybean Oil Index	-1.5115	-1.4189	1.8248
Soybean Future	-1.5155	-1.4189	1.8450
Soybean Index	-1.5088	-1.4188	1.7667
Oats Future	-1.5155	-1.4189	2.0040
Oats Index	-1.5028	-1.4187	2.2831
Corn Future	-1.4993	-1.4189	1.8691
Corn Index	-1.5125	-1.4189	1.8018
Canola Future	-1.5025	-1.4189	1.8116
Canola Index	-1.5061	-1.4188	1.7889
Sugar Future	-1.5064	-1.4189	1.8248
Sugar Index	-1.5125	-1.4188	1.8382
Orange Juice Future	-1.5052	-1.4189	1.9305
Orange Juice Index	-1.5105	-1.4186	1.8519
Cocoa Future	-1.5155	-1.4189	1.8519
Cocoa Index	-1.5154	-1.4188	1.7857
Coffee Future	-1.5007	-1.4189	1.8553
Coffee Index	-1.5111	-1.4188	1.8282
Aluminum Future	-1.5155	-1.4188	1.0274
Aluminum Index	-1.4948	-1.4188	1.5552
Gold Future	-1.4842	-1.4189	1.6978
Gold Index	-1.5155	-1.4189	1.7241
Copper Future	-1.5103	-1.4189	1.6779
Copper Index	-1.5136	-1.4189	1.8083
Palladium Future	-1.5128	-1.4189	1.6447
Palladium Index	-1.5154	-1.4186	1.8182
Platinum Future	-1.5062	-1.4189	1.8182
Platinum Index	-1.5044	-1.4189	1.7606
Lean Hogs Future	-1.4259	-1.4189	1.8149
Lean Hogs Index	-1.5155	-1.4189	1.8519
Feeder Cattle Future	-1.4944	-1.4189	1.7182
Feeder Cattle Index	-1.5037	-1.4188	1.6779
Crude Oil Future	-1.5059	-1.4189	1.8018
Crude Oil Index	-1.5217	-1.4188	1.6750

Table 6.8: Log-likelihood comparison for $\alpha = 2$ and estimated α's

Product	AIC_BIC_HQIC	Best AR-I-MA Lags	Corresponding ARIMA Parameters
Wheat	16409.52026	[2, 1, 0]	[0.0811 0.0075] []
Soybean Oil	2406.835662	[2, 0, 0]	[1.0196 -0.0380] []
Soybean	18361.21968	[2, 1, 0]	[-0.0385 0.0051] []
Oats	14158.94361	[2, 1, 1]	[-0.7380 0.0922] [0.7869]
Corn	14373.87495	[4, 0, 2]	[1.3767 -1.3634 0.9807 -0.0436] [-0.3942 1.0817]
Canola	14445.53524	[4, 1, 1]	[0.9971 0.0205 0.0327 -0.0844] [-0.9999]
Sugar	1107.097312	[2, 1, 0]	[0.0484 -0.0506] []
Orange Juice	12119.73917	[2, 1, 0]	[0.0896 -0.0173] []
Cocoa	23915.95993	[2, 1, 0]	[-0.0072 0.0422] []
Coffee	10920.96411	[2, 1, 0]	[0.0389 0.0297] []
Aluminum	20304.80526	[2, 2, 1]	[0.0026 0.0078] [-0.9998]
Gold	17716.73324	[2, 1, 0]	[-0.0166 -0.0136] []
Copper	26564.95893	[2, 2, 1]	[-0.0527 0.0175] [-0.9999]
Palladium	18210.23595	[2, 2, 1]	[-0.0074 -0.0179] [-0.9999]
Platinum	18640.24702	[2, 1, 0]	[-0.0042 0.0230] []
Lean Hogs	8727.696656	[2, 0, 1]	[1.9828 -0.9836] [-0.9824]
Feeder Cattle	9934.99969	[2, 1, 0]	[0.0715 0.0115] []
Crude Oil	7270.214837	[2, 1, 0]	[-0.0785 -0.0270] []

Table 6.9: ARIMA model estimation results for future data sets best AR-I-MA lags gives the lags for AR, MA terms and if the first difference is taken. The corresponding parameters for AR and MA terms are presents in the two square brackets in the last column.

6.3 Source Code

```
# Nina Ni (2019@Liverpool)

import pandas as pd
import numpy as np
from scipy import stats
from sklearn.metrics import mean_squared_error

# performance the ADJ and KPSS test for giving data sets
stats_tests = False
if stats_tests == True:
    # perform ADF test
    from statsmodels.tsa.stattools import adfuller
    from statsmodels.tsa.stattools import kpss

    for this_product in source_index_df:
        # print
        print('The␣ADF␣and␣KPSS␣tests␣for␣', str(this_product), '␣
            index␣are:')
        # obtain current product prices
        this_product_array = source_index_df[this_product].values
        # apply adf / kpss test to original data, log data and
            increment data
        for i in [1,10,50]:
            # ADF
            print('The␣p-values␣for␣ADF␣tests␣are:␣%.4f␣%.4f␣%.4f'
                %
                (adfuller(this_product_array,maxlag=i, regression='
                    ct')[1],
                adfuller(np.log(this_product_array),maxlag=i,
                    regression='ct')[1],
                adfuller(np.diff(this_product_array),maxlag=i,
                    regression='ct')[1]),
                '(original,␣log,␣increment)␣for␣lag␣', str(i),'.')
        for i in [1,10,50]:
            # KPSS
            print('The␣p-values␣for␣KPSS␣tests␣are:␣%.4f␣%.4f␣%.4f'
                %
                (kpss(this_product_array,lags=i, regression='ct')
                    [1],
                kpss(np.log(this_product_array),lags=i, regression=
                    'ct')[1],
                kpss(np.diff(this_product_array),lags=i, regression
                    ='ct')[1]),
                '(original,␣log,␣increment)␣for␣lag␣', str(i),'.')
```

```
        print('')


    for this_product in source_future_df:
        # print
        print('The ADF and KPSS tests for ', str(this_product), '
            future are:')
        # obtain current product prices
        this_product_array = source_future_df[this_product].values
        # apply adf / kpss test to original data, log data and
            increment data
        for i in [1,10,50]:
            # ADF
            print('The p-values for ADF tests are: %.4f %.4f %.4f'
                %
                (adfuller(this_product_array,maxlag=i, regression='
                    ct')[1],
                adfuller(np.log(this_product_array),maxlag=i,
                    regression='ct')[1],
                adfuller(np.diff(this_product_array),maxlag=i,
                    regression='ct')[1]),
                '(original, log, increment) for lag ', str(i),'.')
        for i in [1,10,50]:
            # KPSS
            print('The p-values for KPSS tests are: %.4f %.4f %.4f'
                %
                (kpss(this_product_array,lags=i, regression='ct')
                    [1],
                kpss(np.log(this_product_array),lags=i, regression=
                    'ct')[1],
                kpss(np.diff(this_product_array),lags=i, regression
                    ='ct')[1]),
                '(original, log, increment) for lag ', str(i),'.')

        print('')


## use likelihood ratio test to test if alpha-stable is better

# find log-diff of the data and test if it is normal
# loop over products and conduct the normal test
for i = 1:length(all_price_data)
    this_data = data_cleansing(all_price_data{i,2});
    all_price_data{i,1}
    this_data = diff(log(this_data));
```

```
    # working on normal tests
    this_data_stdN = (this_data - mean(this_data)) / std(this_data)
        ;
    # Anderson-Darling test
    [ad_test_h(i),ad_test_p(i)] = adtest(this_data_stdN);
    # One-sample Kolmogorov-Smirnov test
    [ks_test_h(i),ks_test_p(i)] = kstest(this_data_stdN);
    # Loglikelihood for normal and alpha stable distribution
    # normal
    tmp = normpdf(this_data_stdN);
    Nlog(i) = mean(log(tmp(tmp > 0)));
    # alpha
    pd1 = makedist('Stable','alpha',min(2,1 / mean_GHE(i)),'beta'
        ,0,'gam',1,'delta',0);
    Alog(i) = mean(log(pdf(pd1,this_data_stdN)));
    # Likelihood ratio test of model specification
    [LR_test_h(i),LR_test_p(i)] = lratiotest(Alog(i),Nlog(i),1);
end

Test_Table = table({all_price_data{:,1}}',(1./mean_GHE)',ad_test_h
    ',ad_test_p',...
    ks_test_h',ks_test_p',Alog',Nlog', LR_test_h',LR_test_p');
Test_Table.Properties.VariableNames = {'Product','alpha','AD_h','
    AD_p',...
    'KS_h','KS_p','LH_a','LH_N','LR_h','LR_p'};
writetable(Test_Table, 'Test_Table.csv')


# use arima model to perform a one-stop forecast
ARIMA_forecast = False
if ARIMA_forecast == True:
    from statsmodels.tsa.arima_model import ARIMA
    import itertools
    from miniutils import progress_bar
    import datetime
    import os
    import matplotlib.pyplot as plt
    # arima estimation
    def arima_fit(data_series, ar, d, ma):
        try:
            model = ARIMA(data_series, order = [ar, d, ma])
            model_fit = model.fit(disp=0)
            AIC_BIC_HQIC = model_fit.aic + model_fit.bic +
                model_fit.hqic
        except:
            model_fit = 'Fitting error!'
```

```
            AIC_BIC_HQIC = np.nan

        return {'order':[ar,d,ma], 'model_fit':model_fit,
                'AIC_BIC_HQIC': AIC_BIC_HQIC}

    # this function finds the best arima model via compare
        information criteria
    def find_best_arima(data, ar_ranges = [0,1,2,4,8], ma_ranges =
        [0,1,2,4,6], integral_ranges = [1,2,3]):
        # build all arima estimation parameter combinations
        parameters = list(itertools.product(ar_ranges,
            integral_ranges, ma_ranges))
        # run the models in loop or parallel
        result = []
        for this_set in parameters:
            result.append(arima_fit(data_series = data, ar =
                this_set[0], d= this_set[1], ma = this_set[2]))
        # put together all result
        full_result = pd.DataFrame(result)
        # sort by the smallest AIC + BIC + HQIC
        full_result = full_result.sort_values('AIC_BIC_HQIC', axis
            =0, ascending=True)
        # return the best model
        best_model = full_result.iloc[0]
        return {'Product': data.name,
                'Best_AR-I-MA_lags': best_model.order,
                'Corresponding_ARIMA_paramters': str(best_model.
                    model_fit.arparams) + str(best_model.model_fit.
                    maparams),
                'AIC_BIC_HQIC': best_model.AIC_BIC_HQIC,
                'Best_model': best_model
                }

    # function to get the forecast done for given product
    def forecast_product(product_name, source_df):
        try:
            # print('Forecasting by ARIMA for ', str(product_name),
                ' stars...')
            # obtain current product prices
            this_product_array = source_df[product_name]
            training_series = \
                this_product_array[this_product_array.index <=
                    datetime.datetime(year=2017, month=12, day=31)]
            training_series = \
                training_series[training_series.index > datetime.
                    datetime(year=2015, month=1, day=1)]
```

```
        # find the best model with all historical data till end
            2017
        best_model = \
            find_best_arima(training_series, ar_ranges=[2, 4,
                8], ma_ranges=[0, 1, 2], integral_ranges
                =[0,1,2])
        # making one step forecast
        forecast_dates = this_product_array.index[
            this_product_array.index > datetime.datetime(year
                =2017, month=12, day=31)]
        forecast_data = pd.DataFrame(this_product_array)
        this_data_series = training_series

        for this_day in forecast_dates:
            # adding one day data to the model
            this_arima = ARIMA(this_data_series, order=
                best_model['Best AR-I-MA lags'])
            this_arima_fit = this_arima.fit(disp=0)
            forecast_data.loc[this_day, 'TS_Forecast'] =
                this_arima_fit.forecast()[0]
            this_data_series[this_day] = this_product_array[
                this_day]

        # write results to file
        if not (os.path.isdir('./output')):
            os.mkdir('./output')
        forecast_data[forecast_data.index > datetime.datetime(
            year=2017, month=12, day=31)].\
            to_csv('./output/'+str(product_name) + '.csv')
        forecast_data[forecast_data.index > datetime.datetime(
            year=2017, month=12, day=31)].plot()
        plt.show()

        return {'Product': best_model['Product'],
                'Best AR-I-MA lags': best_model['Best AR-I-MA
                    lags'],
                'Corresponding ARIMA paramters': best_model['
                    Corresponding ARIMA paramters'],
                'AIC_BIC_HQIC': best_model['AIC_BIC_HQIC'],
                }


## NARX Neural Network Forecast (Matlab Neural Network Toolbox)
# Time steps need to be forecast
Predict_Time = 180;
# Number of time delay in NN
```

```
Time_Delay = 2;

# NNinput - input time series.
m = Predict_Time / Time_Delay;
NN_Result = zeros(Predict_Time,1);

for q = 1:m
NNinput = Group1(1 : end-Time_Delay*m + (q-1)*Time_Delay,:);
X = tonndata(NNinput,false,false);
T = tonndata(r(1 : end-Time_Delay*m + (q-1)*Time_Delay),false,false
    );

# Choose a Training Function 'trainlm' is usually fastest.
trainFcn = 'trainlm';

# Create a Nonlinear Autoregressive Network with External Input
inputDelays = 1:Time_Delay;
feedbackDelays = 1:Time_Delay;

# Choose the optimal number of hidden layers
for p = 1:5
    [ nets{p}, performance_option(p)] = NARX_Optimal...
    ( p, X, T, inputDelays,feedbackDelays,trainFcn );
end

p = find(logical(performance_option = min(performance_option)),1);

net = nets{p};
[x,xi,ai,t] = preparets(net,X,{},T);

fprintf('\nWe have optimal layer(s) is %.i.',p)

# Multi-step Prediction. The function CLOSELOOP replaces the
    feedback input with a direct connection from the outout layer.
netc = closeloop(net);
netc.name = [net.name ' - Closed Loop'];
[xc,xic,aic,tc] = preparets(netc,X,{},T);
yc = netc(xc,xic,aic);
closedLoopPerformance = perform(net,tc,yc);

# Sometimes it is useful to simulate a network in open-loop form
    for as long as there is known output data, and then switch to
    closed-loop form to perform multistep prediction while
    providing only the external input.
# Here the input series and target series are used to simulate the
    network in open-loop form, taking advantage of the higher
```

```
    accuracy that providing the target series produces:

numTimesteps = size(x,2);
knownOutputTimesteps = 1:(numTimesteps-Time_Delay);
predictOutputTimesteps = (numTimesteps-Time_Delay+1):numTimesteps;
X1 = X(:,knownOutputTimesteps);
T1 = T(:,knownOutputTimesteps);
[x1,xio,aio] = preparets(net,X1,{},T1);
[y1,xfo,afo] = net(x1,xio,aio);

# Next the the network and its final states will be converted to
    closed-loop form to make predictions.
x2 = X(1,predictOutputTimesteps);
[netc,xic,aic] = closeloop(net,xfo,afo);
[y2,xfc,afc] = netc(x2,xic,aic);
multiStepPerformance = perform(net,T(1,predictOutputTimesteps),y2);


## plot out all results and perform the tests

class stats_tests():

    def __init__(self, real, est):
        self.real = real
        self.est = est

    def r_squared(self):
        slope, intercept, r_value, p_value, std_err = stats.
            linregress(self.real, self.est)
        return r_value**2

    def NMSR(self):
        # Normalized Mean Squared Error
        return mean_squared_error(self.real, self.est) / (np.std(
            self.real) ** 2)

    def MAPE(self):
        # Mean Absolute Percentage Error
        return np.nanmean(np.abs(self.est - self.real) / self.real)

    def DC(self):
        # Directional Change
        return np.nanmean(np.diff(self.real) * (self.est[1:] - self
            .real[:-1]) >= 0)
```

```
if True:
    import datetime
    import matplotlib.pyplot as plt
    import matplotlib.dates as mdates
    import pathlib
    # list all files from the machine learning results folder
    all_results = list(pathlib.Path('./output').glob('*.csv'))
    test_results = []
    for this_file in all_results:
        this_product = str(this_file).split('/')[1][:-4]
        if this_product == 'model_selection':
            pass
        else:
            print('Plotting ', str(this_file), ' for product: ',
                str(this_product))
            this_pd = pd.read_csv(this_file, index_col= 'Dates.1')

            # Set the locator
            locator = mdates.MonthLocator()  # every month
            # Specify the format - %b gives us Jan, Feb...
            fmt = mdates.DateFormatter('%b')
            plt.figure(figsize=(8, 5),dpi=160)

            this_pd[this_product].plot(color='#979797', linewidth
                =3)

            this_pd['MLP'].plot(color='#ED8F12', linewidth=1)
            this_pd['NARX_T'].plot(color='#9E12B6', linewidth=1)
            this_pd['NARX_F'].plot(color='#1AB716', linewidth=1)
            this_pd['LSTM'].plot(color='#1FB8DA', linewidth=1)
            this_pd['TS_Forecast'].plot(color='#F10E1F', linewidth
                =1)

            plt.xticks([0,22,44,66,88,110,132,154,175],
                       ['Jan', 'Feb', 'Mar', 'Apr', 'May','Jue', '
                           Jul','Aug'])

            plt.legend(('Observations', 'MLP', 'NARX_Time','
                NARX_Frequency','LSTM','ARIMA'))
            plt.title(r"$\bf{" + this_product + "}$")


            plt.xlabel('Date (Year 2018)')
            plt.ylabel('Price')
            plt.grid(True)
            plt.savefig('./output/' + this_product + '.eps', format
```

```
            ='eps')

        # perform the test
        st = stats_tests(this_pd[this_product].values, this_pd[
            'TS_Forecast'].values)
        this_result = {
            'Product': this_product,
            'NMSE': st.NMSR(),
            'MAPE': st.MAPE(),
            'R2': st.r_squared(),
            'Dstst': st.DC(),
        }
        test_results.append(this_result)

pd.DataFrame(test_results).to_csv('./stats_test.csv', index =
    False)
```

Bibliography

[1] Akaike, H. (1974). A new look at the statistical model identification. *IEEE transactions on automatic control 19*(6), 716–723.

[2] Akintola, K., B. Alese, and A. Thompson (2011). Time series forecasting with neural network: A case study of stock prices of intercontinental bank nigeria. *IJRRAS Dec2011*.

[3] Arthur, W. B. (2018). Asset pricing under endogenous expectations in an artificial stock market. In *The economy as an evolving complex system II*, pp. 31–60. CRC Press.

[4] Avram, F. and M. S. Taqqu (2000). Robustness of the R/S statistic for fractional stable noises. *Statistical inference for stochastic processes 3*(1-2), 69–83.

[5] Baek, J., G. J. McLachlan, and L. K. Flack (2010). Mixtures of factor analyzers with common factor loadings: Applications to the clustering and visualization of high-dimensional data. *IEEE transactions on pattern analysis and machine intelligence 32*(7), 1298–1309.

[6] Baillie, R. T. (1996). Long memory processes and fractional integration in econometrics. *Journal of econometrics 73*(1), 5–59.

[7] Baillie, R. T., C.-F. Chung, M. A. Tieslau, et al. (1996). Analysing inflation by the fractionally integrated arfima-garch model. *Journal of applied econometrics 11*(1), 23–40.

[8] Banfield, J. D. and A. E. Raftery (1993). Model-based gaussian and non-gaussian clustering. *Biometrics*, 803–821.

[9] Barkoulas, J., W. C. Labys, and J. Onochie (1997). Fractional dynamics in international commodity prices. *Journal of Futures Markets: Futures, Options, and Other Derivative Products 17*(2), 161–189.

[10] Barkoulas, J. T., W. C. Labys, and J. I. Onochie (1999). Long memory in futures prices. *Financial Review 34*(1), 91–100.

[11] Barndorff-Nielsen, O. E. and N. Shephard (2001). Modelling by lévy processess for financial econometrics. In *Lévy processes*, pp. 283–318. Springer.

[12] Beran, J. (2017). *Statistics for long-memory processes*. Routledge.

[13] Bollerslev, T., R. Y. Chou, and K. F. Kroner (1992). Arch modeling in finance: A review of the theory and empirical evidence. *Journal of econometrics 52*(1-2), 5–59.

[14] Bollerslev, T., R. F. Engle, and D. B. Nelson (1994). Arch models. *Handbook of econometrics 4*, 2959–3038.

[15] Bouchaud, J.-P. and M. Potters (2001). More stylized facts of financial markets: leverage effect and downside correlations. *Physica A: Statistical Mechanics and its Applications 299*(1-2), 60–70.

[16] Bouveyron, C. and J. Jacques (2011). Model-based clustering of time series in group-specific functional subspaces. *Advances in Data Analysis and Classification 5*(4), 281–300.

[17] Box, G. E., G. M. Jenkins, G. C. Reinsel, and G. M. Ljung (2015). *Time series analysis: forecasting and control*. John Wiley & Sons.

[18] Brockwell, P. J. and R. A. Davis (2016). *Introduction to time series and forecasting*. springer.

[19] Brooks, R. A. (1986). Achieving artificial intelligence through building robots. Technical report, MASSACHUSETTS INST OF TECH CAMBRIDGE ARTIFICIAL INTELLIGENCE LAB.

[20] Brunelli, R. and T. Poggio (1993). Face recognition: Features versus templates. *IEEE transactions on pattern analysis and machine intelligence 15*(10), 1042–1052.

[21] Caliński, T. and J. Harabasz (1974). A dendrite method for cluster analysis. *Communications in Statistics-theory and Methods 3*(1), 1–27.

[22] Cao, L.-J. and F. E. H. Tay (2003). Support vector machine with adaptive parameters in financial time series forecasting. *IEEE Transactions on neural networks 14*(6), 1506–1518.

[23] Chiou, J.-M. and P.-L. Li (2007). Functional clustering and identifying substructures of longitudinal data. *Journal of the Royal Statistical Society: Series B (Statistical Methodology) 69*(4), 679–699.

[24] Comte, F. and E. Renault (1996). Long memory continuous time models. *Journal of Econometrics 73*(1), 101–149.

[25] Cont, R. (2001). Empirical properties of asset returns: stylized facts and statistical issues.

[26] Cont, R. (2005). Long range dependence in financial markets. In *Fractals in Engineering*, pp. 159–179. Springer.

[27] Cont, R., J.-P. Bouchaud, and M. Potters (1997). Scaling in financial data: Stable laws and beyond. *Scale Invariance and Beyond, Dubrulle, B., Graner, F., and Sornette, D., eds*.

[28] Coolen, F. (2004). On the use of imprecise probabilities in reliability. *Quality and Reliability Engineering International 20*(3), 193–202.

[29] Cootner, P. H. (1962). Stock prices: Ramdom vs. systematic changes. *Industrial Management Review (pre-1986) 3*(2), 24.

[30] Cowles, A. (1944). Stock market forecasting. *Econometrica, Journal of the Econometric Society*, 206–214.

[31] Crato, N. and P. Rothman (1994). Fractional integration analysis of long-run behavior for us macroeconomic time series. *Economics Letters 45*(3), 287–291.

[32] Crouse, M., R. D. Nowak, and R. G. Baraniuk (1998). Wavelet-based statistical signal processing using hidden markov models. *IEEE Transactions on Signal Processing 46*(4), 886–902.

[33] Delaigle, A. and P. Hall (2010). Defining probability density for a distribution of random functions. *The Annals of Statistics*, 1171–1193.

[34] Ding, Z., C. W. Granger, and R. F. Engle (1993a). A long memory property of stock market returns and a new model. *Journal of empirical finance 1*(1), 83–106.

[35] Ding, Z., C. W. Granger, and R. F. Engle (1993b). A long memory property of stock market returns and a new model. *Journal of empirical finance 1*(1), 83–106.

[36] Doukhan, P., G. Oppenheim, and M. Taqqu (2002). *Theory and applications of long-range dependence*. Springer Science & Business Media.

[37] Enke, D. and S. Thawornwong (2005). The use of data mining and neural networks for forecasting stock market returns. *Expert Systems with applications 29*(4), 927–940.

[38] Fama, E. F. (1970). Efficient capital markets: A review of theory and empirical work. *The journal of Finance 25*(2), 383–417.

[39] Ferraty, F. and P. Vieu (2006). *Nonparametric functional data analysis: theory and practice*. Springer Science & Business Media.

[40] Friedman, M. (1953). The case for flexible exchange rates.

[41] Geary, D. C. (2005). *The origin of mind: Evolution of brain, cognition, and general intelligence*. American Psychological Association.

[42] Gers, F. A., D. Eck, and J. Schmidhuber (2002). Applying lstm to time series predictable through time-window approaches. In *Neural Nets WIRN Vietri-01*, pp. 193–200. Springer.

[43] Gers, F. A., J. Schmidhuber, and F. Cummins (1999). Learning to forget: Continual prediction with lstm.

[44] Gordon, A. D. (1996). Null models in cluster validation. In *From data to knowledge*, pp. 32–44. Springer.

[45] Granger, C. W. (1966). The typical spectral shape of an economic variable. *Econometrica: Journal of the Econometric Society*, 150–161.

[46] Granger, C. W. and R. Joyeux (1980). An introduction to long-memory time series models and fractional differencing. *Journal of time series analysis 1*(1), 15–29.

[47] Green, P. J. and B. W. Silverman (1993). *Nonparametric regression and generalized linear models: a roughness penalty approach*. CRC Press.

[48] Greene, M. T. and B. D. Fielitz (1977). Long-term dependence in common stock returns. *Journal of Financial Economics 4*(3), 339–349.

[49] Grossman, S. J. and J. E. Stiglitz (1980). On the impossibility of informationally efficient markets. *The American economic review 70*(3), 393–408.

[50] Guillaume, D. M., M. M. Dacorogna, R. R. Davé, U. A. Müller, R. B. Olsen, and O. V. Pictet (1997). From the bird's eye to the microscope: A survey of new stylized facts of the intra-daily foreign exchange markets. *Finance and stochastics 1*(2), 95–129.

[51] Hamilton, J. D. (1989). A new approach to the economic analysis of nonstationary time series and the business cycle. *Econometrica: Journal of the Econometric Society*, 357–384.

[52] Harrison, B. and D. Paton (2004). Transition, the evolution of stock market efficiency and entry into eu: the case of romania. *Economics of Planning 37*(3-4), 203.

[53] Harvey, A. C. and A. Jaeger (1993). Detrending, stylized facts and the business cycle. *Journal of applied econometrics 8*(3), 231–247.

[54] Haubrich, J. G. and A. W. Lo (1989). The sources and nature of long-term memory in the business cycle. Technical report, National Bureau of Economic Research.

[55] Henry, O. T. (2002). Long memory in stock returns: some international evidence. *Applied Financial Economics 12*(10), 725–729.

[56] Hochreiter, S. and J. Schmidhuber (1997). Long short-term memory. *Neural computation 9*(8), 1735–1780.

[57] Hosking, J. R. (1981). Fractional differencing. *Biometrika 68*(1), 165–176.

[58] Hotelling, H. (1933). Analysis of a complex of statistical variables into principal components. *Journal of educational psychology 24*(6), 417.

[59] Huang, S.-C. and T.-K. Wu (2008). Combining wavelet-based feature extractions with relevance vector machines for stock index forecasting. *Expert Systems 25*(2), 133–149.

[60] Hurst, H. E., R. P. Black, and Y. Simaika (1965). *Long-term storage: an experimental study*. Constable.

[61] Jacques, J. and C. Preda (2013). Funclust: A curves clustering method using functional random variables density approximation. *Neurocomputing 112*, 164–171.

[62] Jacques, J. and C. Preda (2014). Model-based clustering for multivariate functional data. *Computational Statistics & Data Analysis 71*, 92–106.

[63] James, G. M. and C. A. Sugar (2003). Clustering for sparsely sampled functional data. *Journal of the American Statistical Association 98*(462), 397–408.

[64] Jolliffe, I. (2002). *Principal component analysis*. Wiley Online Library.

[65] Kaastra, I. and M. Boyd (1996). Designing a neural network for forecasting financial and economic time series. *Neurocomputing 10*(3), 215–236.

[66] Kalpakis, K., D. Gada, and V. Puttagunta (2001). Distance measures for effective clustering of arima time-series. In *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*, pp. 273–280. IEEE.

[67] Karhunen, K. (1946). Zur spektraltheorie stochastischer prozesse.

[68] Kay, S. M. (1988). *Modern Spectral Estimation: Theory and Application*.

[69] Kim, K.-j. (2003). Financial time series forecasting using support vector machines. *Neurocomputing 55*(1-2), 307–319.

[70] Kim, S. and H. Kim (2016). A new metric of absolute percentage error for intermittent demand forecasts. *International Journal of Forecasting 32*(3), 669–679.

[71] Kirman, A. and G. Teyssiere (2002). Microeconomic models for long memory in the volatility of financial time series. *Studies in Nonlinear Dynamics & Econometrics 5*(4).

[72] Kogan, J. (2007). *Introduction to clustering large and high-dimensional data*. Cambridge University Press.

[73] Laird, J. E., A. Newell, and P. S. Rosenbloom (1987). Soar: An architecture for general intelligence. *Artificial intelligence 33*(1), 1–64.

[74] Lang, K. J., A. H. Waibel, and G. E. Hinton (1990). A time-delay neural network architecture for isolated word recognition. *Neural networks 3*(1), 23–43.

[75] LeBaron, B. (2001). Evolution and time horizons in an agent-based stock market. *Macroeconomic Dynamics 5*(02), 225–254.

[76] LeRoy, S. F. and R. D. Porter (1981). The present-value relation: Tests based on implied variance bounds. *Econometrica: Journal of the Econometric Society*, 555–574.

[77] Liao, T. W. (2005). Clustering of time series data—a survey. *Pattern recognition 38*(11), 1857–1874.

[78] Liu, M. (2000). Modeling long memory in stock market volatility. *Journal of Econometrics 99*(1), 139–171.

[79] Lo, A. (2002). Fat tails, long memory, and the stock market since the 1960's', economic notes, 26 (2), 213-46. *INTERNATIONAL LIBRARY OF CRITICAL WRITINGS IN ECONOMICS 146*(2), 221–254.

[80] Lo, A. W. (1989). Long-term memory in stock market prices. Technical report, National Bureau of Economic Research.

[81] Loève, M. (1946). Fonctions aléatoires à décomposition orthogonale exponentielle. *La Revue Scientifique 84*, 159–162.

[82] Lux, T. and M. Marchesi (2000). Volatility clustering in financial markets: a microsimulation of interacting agents. *International journal of theoretical and applied finance 3*(04), 675–702.

[83] Maadooliat, M., J. Z. Huang, and J. Hu (2015). Integrating data transformation in principal components analysis. *Journal of Computational and Graphical Statistics 24*(1), 84–103.

[84] Malhotra, P., L. Vig, G. Shroff, and P. Agarwal (2015). Long short term memory networks for anomaly detection in time series. In *Proceedings*, pp. 89. Presses universitaires de Louvain.

[85] Mandelbrot, B. (1965). Une classe de processus stochastiques homothetiques a soi-application a la loi climatologique de h. e. hurst. *COMPTES RENDUS HEBDOMADAIRES DES SEANCES DE L ACADEMIE DES SCIENCES 260*(12), 3274–+.

[86] Mandelbrot, B. B. (1971). When can price be arbitraged efficiently? a limit to the validity of the random walk and martingale models. *The Review of Economics and Statistics*, 225–236.

[87] Mandelbrot, B. B. (1983). *The fractal geometry of nature*, Volume 173. Macmillan.

[88] Mandelbrot, B. B. (1997). The variation of certain speculative prices. In *Fractals and scaling in finance*, pp. 371–418. Springer.

[89] Mandelbrot, B. B. (2013). *Fractals and scaling in finance: Discontinuity, concentration, risk. Selecta volume E*. Springer Science & Business Media.

[90] Mandelbrot, B. B. and J. W. Van Ness (1968). Fractional brownian motions, fractional noises and applications. *SIAM review 10*(4), 422–437.

[91] Mandelbrot, B. B. and J. R. Wallis (1968). Noah, joseph, and operational hydrology. *Water resources research 4*(5), 909–918.

[92] Manning, C. D., C. D. Manning, and H. Schütze (1999). *Foundations of statistical natural language processing*. MIT press.

[93] Mardia, K. V. (1970). Measures of multivariate skewness and kurtosis with applications. *Biometrika 57*(3), 519–530.

[94] McCleary, R., R. A. Hay, E. E. Meidinger, and D. McDowall (1980). *Applied time series analysis for the social sciences*. Sage Publications Beverly Hills, CA.

[95] Milligan, G. W. and M. C. Cooper (1986). A study of the comparability of external criteria for hierarchical cluster analysis. *Multivariate Behavioral Research 21*(4), 441–458.

[96] Mills, T. C. (1993). Is there long-term memory in uk stock returns? *Applied Financial Economics 3*(4), 303–306.

[97] Moré, J. J. (1978). The levenberg-marquardt algorithm: implementation and theory. In *Numerical analysis*, pp. 105–116. Springer.

[98] Muzy, J.-F., J. Delour, and E. Bacry (2000). Modelling fluctuations of financial time series: from cascade process to stochastic volatility model. *The European Physical Journal B-Condensed Matter and Complex Systems 17*(3), 537–548.

[99] Peters, E. E. (1994). *Fractal market analysis: applying chaos theory to investment and economics*, Volume 24. John Wiley & Sons.

[100] Porrà, J. M. (2006). The (mis) behavior of markets. *Journal of Statistical Physics 122*(2), 373–375.

[101] Priddy, K. L. and P. E. Keller (2005). *Artificial neural networks: an introduction*, Volume 68. SPIE Press.

[102] Ramsay, J. O. (2006). *Functional data analysis*. Wiley Online Library.

[103] Ramsay, J. O. and B. W. Silverman (2002). *Applied functional data analysis: methods and case studies*, Volume 77. Citeseer.

[104] Robinson, P. M. (2003). *Time series with long memory*. Oxford University Press.

[105] Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review 65*(6), 386.

[106] Rumelhart, D. E., G. E. Hinton, and R. J. Williams (1988). Learning representations by back-propagating errors. *Cognitive modeling 5*(3), 1.

[107] Sahami, M., S. Dumais, D. Heckerman, and E. Horvitz (1998). A bayesian approach to filtering junk e-mail. In *Learning for Text Categorization: Papers from the 1998 workshop*, Volume 62, pp. 98–105.

[108] Samorodnitsky, G. (2007). Long range dependence. *Foundations and Trends® in Stochastic Systems 1*(3), 163–257.

[109] Schwarz, G. et al. (1978). Estimating the dimension of a model. *The annals of statistics 6*(2), 461–464.

[110] Serban, N. and L. Wasserman (2005). Cats: clustering after transformation and smoothing. *Journal of the American Statistical Association 100*(471), 990–999.

[111] Sugar, C. A. and G. M. James (2003). Finding the number of clusters in a dataset: An information-theoretic approach. *Journal of the American Statistical Association 98*(463), 750–763.

[112] Sundin, T. and P. Stoica (1998). On nonparametric spectral estimation. In *9th European Signal Processing Conference (EUSIPCO 1998)*, Volume 18, pp. 169–181.

[113] Sze, V., Y.-H. Chen, J. Emer, A. Suleiman, and Z. Zhang (2017). Hardware for machine learning: Challenges and opportunities. In *2017 IEEE Custom Integrated Circuits Conference (CICC)*, pp. 1–8. IEEE.

[114] Tarpey, T. and K. K. Kinateder (2003). Clustering functional data. *Journal of classification 20*(1), 093–114.

[115] Thorndike, R. L. (1953). Who belongs in the family? *Psychometrika 18*(4), 267–276.

[116] Tieslau, M. A. (1992). *Strongly dependent economic time series: Theory and applications*.

[117] Turvey, C. G. (2007). A note on scaled variance ratio estimation of the Hurst exponent with application to agricultural commodity prices. *Physica A: Statistical Mechanics and its Applications 377*(1), 155–165.

[118] Urrutia, J. L. (1995). Tests of random walk and market efficiency for latin american emerging equity markets. *Journal of financial research 18*(3), 299–309.

[119] Werbos, P. (1974). Beyond regression: New tools for prediction and analysis in the behavioral sciences. *Ph. D. dissertation, Harvard University*.

[120] White, H. (1988). Economic prediction using neural networks: The case of ibm daily stock returns.

[121] Whittle, P. (1954). On stationary processes in the plane. *Biometrika*, 434–449.

[122] Working, H. (1949). The investigation of economic expectations. *The American Economic Review*, 150–166.

[123] Worthington, A. C. and H. Higgs (2006). Evaluating financial development in emerging capital markets with efficiency benchmarks.

[124] Zhang, D., Q. Jiang, and X. Li (2004). Application of neural networks in financial data mining. In *International Conference on Computational Intelligence*, pp. 392–395. Citeseer.